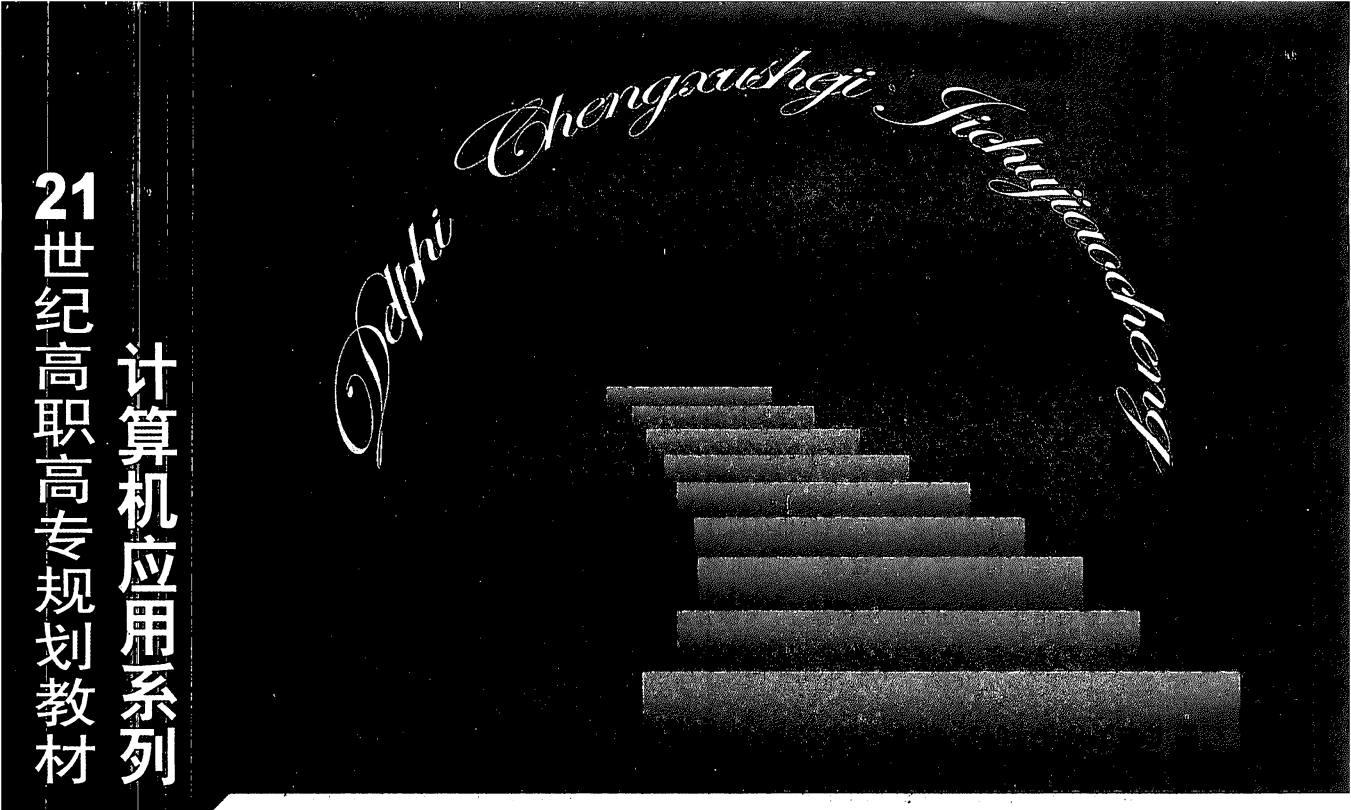


21世纪高职高专规划教材 计算机应用系列

何定华　易海胜　编著

Delphi 程序设计基础教程

清华大学出版社
北京

内 容 简 介

本书采用案例式教学的编写方法，详细讲述 Delphi 程序设计。本书内容丰富，包括 Delphi 基础知识、Object Pascal 语言、编程基础（包含程序控制结构、枚举、子界与集合、数组与记录以及过程与函数）、界面设计（包含窗体与基本组件、对话框、菜单、工具栏、状态栏以及图形图像与多媒体）、文件编程、数据库编程（包含数据库编程基础、BDE 数据库应用程序开发、ADO 数据库应用程序开发和报表设计）以及网络编程。本书以数据库为重点，突出 BDE 应用程序设计和 ADO 程序设计，报表设计讲述则较为简单。相信通过本书的学习，读者一定能够快速掌握 Delphi 这门优秀的开发工具，并能轻松开发出实用的软件和数据库应用程序。

本书不仅可以作为高等学校教材，也可以作为社会培训班教材和参考书。对于那些希望快速学会 Delphi 开发工具的初学者，本书也是一本不可多得的好教材。

图书在版编目（CIP）数据

Delphi 程序设计基础教程/何定华，易海胜编著．—北京：清华大学出版社，2008.5
21 世纪高职高专规划教材．计算机应用系列
ISBN 978-7-302-17043-3

Ⅰ．D…　Ⅱ．①何…②易…　Ⅲ．软件工具－程序设计－高等学校：技术学校－教材
Ⅳ．TP311.56

中国版本图书馆 CIP 数据核字（2008）第 018173 号

责任编辑：刘　青　张　景
责任校对：袁　芳
责任印制：何　芊

出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京鑫丰华彩印有限公司
装 订 者：三河市溧源装订厂
经　　销：全国新华书店
开　　本：185×260　　**印　张**：18.75　　**字　数**：431 千字
版　　次：2008 年 5 月第 1 版　　**印　次**：2008 年 5 月第 1 次印刷
印　　数：1～4000
定　　价：26.00 元

前 言

Delphi 是一种优秀的基于 Windows 的应用软件开发工具。它具有可视化的开发环境、简洁明快的编程语言、强大的数据库功能、可扩展的组件库等特点，Delphi 正成为一种越来越成熟的软件开发产品。聪明的程序员使用 Delphi，这是业界广为所知的一句名言，因此 Delphi 正在赢得越来越多的用户。

本书内容丰富，是 Delphi 的入门级教程。无论读者是否具有编程基础与编程思想，本书都是非常合适的教材与参考书。由于长期从事 Delphi 程序设计的教学工作，本教材是根据多年的教学经验编写而成的。本书可以作为高等院校教材和参考书，也可以作为社会培训班教材。教师可以根据学生的实际情况对某些章节进行加强，或者对某些章节进行适当的删减。

本书具有如下特点。

1. 结构合理

本书将 Delphi 基础、Object Pascal 语言等内容介绍得比较详细，很多内容都独立编写成一章，这样可以充分考虑到入门者的实际情况。而对于有基础的读者，也方便他们对某些章节进行合理的取舍。本书例题丰富，讲述清楚，界面美观，力图深入浅出。本书各个章节的先后顺序也经过作者精心设计，读者或者教师还可以根据自己的喜好和习惯，改变本书学习和讲授的原有顺序。例如，可以将分支结构和单选按钮、复选按钮一起讲解，将循环结构和列表框、组合框一起讲解等。

2. 内容丰富

本书讲述了 Delphi 语言的基础知识，包括第 1、2、4～7 章。这些章节对于有编程思想和编程基础的读者可以适当进行一些取舍，教师在讲授的时候课时分配也可以酌情减少。第 3 章讲述窗体和基本组件，舍弃了一些比较复杂的组件，讲述的组件都是常用的和简单的，适合高校学生的实际情况。这些组件的讲述采用案例式的方法，配备有丰富的实例讲解，利于学生接受和消化。第 8～10 章都是界面编程，所开发的软件界面美观是 Delphi 的一大特点。数据库编程占据了本书的较大篇幅。本书讲述了数据库基础、BDE 数据库、ADO 数据库，还简单讲述了使用 Rave 组件制作数据库报表。第 11 章是文件编程。数据库章节包括第 12～15 章，这是本书的重点，读者学过这几个章节之后，一定会有较大的收获。本书第 16 章还简单讲述了几个网络组件，以及如何使用这几个组件进行网络编程。

3. 重点突出

Delphi 的数据库功能非常强大，这是 Delphi 的特点。虽然本书内容丰富，讲述的知识点也比较多，但是本书的重点非常突出。数据库是本书的重点，本书花了大量篇幅讲述数据库知识。第 12 章讲述数据库基础，第 13 章讲述 BDE 数据库编程，第 14 章讲述 ADO 数据库编程。各个章节例题丰富，讲述清楚，不仅有详细的代码，而且还有可视化的工具软件操作，界面清晰。第 15 章也可以理解为数据库章节，它讲述了 Delphi 报表工具 Rave 5 的用法，内容简洁明快，例题讲述非常详细。读者可以一步一步模拟操作，制作数据库报表。

4. 配备教学资源

本书中的每一道例题，作者都保留了原始文件，且每道例题在 Delphi 6. x、Delphi 7. x 和 Delphi 8. x 中都能正常运行（报表程序不能在 Delphi 6. x 中运行除外，Delphi 6. x 中不含 Rave 报表组件）。本书的所有例题源代码都免费赠送给读者，请读者与作者联系或者从清华大学出版社的相关网页上下载。另外，本书配有讲课用演示文稿 PowerPoint 文件，需要的教师也可以从相关网页中下载。本书还配套有《Delphi 程序设计基础教程上机指导及习题解答》，可以配合本书使用，也可以单独使用。

本教材由何定华、易海胜编写。易海胜编写第 1 章，其他章节由何定华编写。

由于时间仓促，加之水平有限，不妥与疏漏之处在所难免，恳请读者批评指正。

联系方式，E-mail：hedinghua2002@163. com，QQ：287125761。

作　者

2007 年 3 月于武汉

计算机精品学习资料大放送

软考官方指定教材及同步辅导书下载 | 软考历年真是解析与答案

软考视频 | 考试机构 | 考试时间安排

Java 一览无余：Java 视频教程 | Java SE | Java EE

.Net 技术精品资料下载汇总：ASP.NET 篇

.Net 技术精品资料下载汇总：C#语言篇

.Net 技术精品资料下载汇总：VB.NET 篇

撼世出击：C/C++编程语言学习资料尽收眼底 电子书+视频教程

Visual C++(VC/MFC)学习电子书及开发工具下载

Perl/CGI 脚本语言编程学习资源下载地址大全

Python 语言编程学习资料(电子书+视频教程)下载汇总

最新最全 Ruby、Ruby on Rails 精品电子书等学习资料下载

数据库精品学习资源汇总：MySQL 篇 | SQL Server 篇 | Oracle 篇

最强 HTML/xHTML、CSS 精品学习资料下载汇总

最新 JavaScript、Ajax 典藏级学习资料下载分类汇总

网络最强 PHP 开发工具+电子书+视频教程等资料下载汇总

UML 学习电子资下载汇总 软件设计与开发人员必备

经典 LinuxCBT 视频教程系列 Linux 快速学习视频教程一帖通

天罗地网：精品 Linux 学习资料大收集(电子书+视频教程) Linux 参考资源大系

Linux 系统管理员必备参考资料下载汇总

Linux shell、内核及系统编程精品资料下载汇总

UNIX 操作系统精品学习资料<电子书+视频>分类总汇

FreeBSD/OpenBSD/NetBSD 精品学习资源索引 含书籍+视频

Solaris/OpenSolaris 电子书、视频等精华资料下载索引

目 录

计算机精品学习资料大放送

软考官方指定教材及同步辅导书下载 | 软考历年真是解析与答案

软考视频 | 考试机构 | 考试时间安排

Java 一览无余：Java 视频教程 | Java SE | Java EE

.Net 技术精品资料下载汇总：ASP.NET 篇

.Net 技术精品资料下载汇总：C#语言篇

.Net 技术精品资料下载汇总：VB.NET 篇

撼世出击：C/C++编程语言学习资料尽收眼底 电子书+视频教程

Visual C++(VC/MFC)学习电子书及开发工具下载

Perl/CGI 脚本语言编程学习资源下载地址大全

Python 语言编程学习资料(电子书+视频教程)下载汇总

最新最全 Ruby、Ruby on Rails 精品电子书等学习资料下载

数据库精品学习资源汇总：MySQL 篇 | SQL Server 篇 | Oracle 篇

最强 HTML/xHTML、CSS 精品学习资料下载汇总

最新 JavaScript、Ajax 典藏级学习资料下载分类汇总

网络最强 PHP 开发工具+电子书+视频教程等资料下载汇总

UML 学习电子资下载汇总 软件设计与开发人员必备

经典 LinuxCBT 视频教程系列 Linux 快速学习视频教程一帖通

天罗地网：精品 Linux 学习资料大收集(电子书+视频教程) Linux 参考资源大系

Linux 系统管理员必备参考资料下载汇总

Linux shell、内核及系统编程精品资料下载汇总

UNIX 操作系统精品学习资料<电子书+视频>分类总汇

FreeBSD/OpenBSD/NetBSD 精品学习资源索引 含书籍+视频

Solaris/OpenSolaris 电子书、视频等精华资料下载索引

第 1 章

Delphi 基础知识

Delphi 是由 Borland(现 Inprise)公司于 1995 年推出的快速应用软件开发工具 RAD (Rapid Application Development)。Delphi 使用了当今世界上最为先进的很多程序开发思想,使用 Delphi 开发软件无疑会大大提高软件开发的效率。

Delphi 到现在为止经历了多代发展历程,每一代产品都伴随 Windows 操作系统升级而升级。本书采用 Delphi 7 作为平台,讲述 Delphi 的相关知识。本书的所有程序在 Delphi 6. x、Delphi 7. x 和 Delphi 8. x 上都运行通过。

1.1 Delphi 7 简介

Delphi 7 是 Windows 系统下的可视化集成开发工具,它提供了强大的可视化组件 VCL(Visual Component Library)功能,使程序员可以快速、高效地开发出 Windows 系统下的应用程序。和其他软件开发工具相比,Delphi 在网络编程、数据库编程、程序界面方面更胜一筹,而且增加了对 Linux 平台下应用程序开发的更强大的支持。

Delphi 7 有 3 个版本,分别是标准版(Standard)、专业版(Professional)和企业版(Enterprise),分别适用于一般用途开发、较高用途开发和进行多层数据库及分布式应用开发。本书采用 Delphi 7 企业版。

1.1.1 Delphi 的特点和优点

1. 可视化的集成开发环境

Delphi 提供了可视化的集成开发环境 IDE(Integrated Development Environment)。可视化环境是指用户在设计程序界面的时候,无须为程序界面编写代码,只需要将相应的组件添加到窗体上,调节其大小和位置即可,Delphi 会自动生成相应的代码。Delphi 的集成开发环境不仅可以非常方便地设计出用户的程序界面,而且程序的界面设计、属性设置、代码编写、程序调试、运行、生成可执行文件等操作都可以在这个集成开发环境中进行,使得软件设计变得非常快捷、高效。

2. 真正的面向对象

面向对象的程序设计 OOP(Object Oriented Programming)是 Delphi 诞生的基础。像 Visual Basic 这样的语言是伪面向对象的,它不支持封装、继承性和多态性等面向对象

```
Label1.Font.Name:='隶书';
```

3. 对象的方法

方法(Method)是封装在对象中的一段代码,用来实现预先规定好的功能。方法是对象能够执行的动作,它由 Delphi 内部定义,用户不用编写代码即可实现某种功能,如 Show(显示)、Move(移动)、Line(画线)。不同的对象有自己的方法集合。在程序中调用方法的语法为:

```
对象.方法(方法需要的参数);
```

以一个 Delphi 对象为例,调用编辑框的方法 SetFocus 让编辑框得到焦点。代码为:

```
Edit1.SetFocus;
```

4. 对象的事件

事件(Event)是作用在对象上,并且能够被对象识别的动作。用户不能建立新的事件,为此 Delphi 提供了大量的事件,以满足程序员的需要。比如,用鼠标单击命令按钮,命令按钮能够识别这个动作,并且程序会立即去执行存放在命令按钮的单击事件过程里面的代码,这样就完成了用户和程序之间的交互。

在面向过程的程序设计中,程序是按照预先设置好的顺序执行的。这就意味着程序与用户之间的交互相当有限。在面向对象的程序设计中引入了一个新的概念,也就是事件驱动机制。程序的执行不再按照某个固定不变的顺序进行。程序代码放在事件过程(Event Procedure)中。一个事件发生后,程序会自动执行对应的事件过程。

事件的触发就是事件是怎样发生的,它有以下 4 种方式。

(1) 用户通过交互方式触发事件。比如,用户单击命令按钮就触发了命令按钮的单击事件。

(2) 时间触发。在 Delphi 中有一个控件 Timer,它有一个属性 Interval,这个控件能够每隔 Interval 毫秒自动触发 OnTimer 事件。

(3) 系统触发。例如,OnCreate 事件在窗体加载到内存的时候自动发生,OnClose 事件在窗体从内存中卸载的时候自动触发。

(4) 可以用代码调用使事件发生。例如,执行 Form1. Close;语句后程序自动触发 OnClose 事件关闭窗体。

1.1.3 类和组件

类和组件是 Delphi 中重要的概念,充分体现了面向对象的基本思想,是 Delphi 可视化编程的基础。

1. Delphi 中的类和组件

Delphi 的组件是由可视化组件库(Visual Component Library,VCL)提供的。这个组件库是一个基类,其中定义了很多子类。VCL 的有些子类被可视化地安排在组件面板上,这就是组件。有的类则是不可见的。VCL 的所有的类都是从 TObject 派生出来的,因此 TObject 又叫做基类。而可视化类(组件)是从 TComponent 类派生出来的,如 TButton、TEdit、TLabel 等。

组件是建立程序界面的基本元素，是可视化编程的基础。正因为有了组件才使得编写 Delphi 应用程序极为方便。组件是具有相同特性的事物的抽象，包括事物静态特性和动态特性的描述，是创建对象的模板。

图 1-1 是 VCL 中主要的类以及它们之间的层次关系。

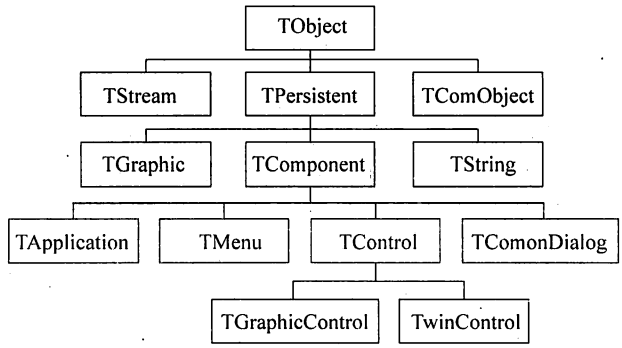


图 1-1　VCL 中主要类之间的层次关系

2. Delphi 中最常见的组件

Delphi 7 共有 32 个组件选项卡，共包括近 400 个组件。表 1-1～表 1-3 给出了 Standard 组件选项卡、Additional 组件选项卡和 Win32 组件选项卡的一些常见组件以及这些组件的简要说明。

表 1-1　Standard 组件选项卡中最常见的组件

组件图标	组 件 名 称	说　　明
	Mainmenu	用于创建下拉式菜单
	PopupMenu	用于创建弹出式菜单
A	Label	标签，用于显示静态的文字
abl	Edit	单行编辑框，用于输入单行文本，可以编辑
	Memo	多行编辑框
OK	Button	普通按钮
	Check	多选按钮，为用户提供选项
	RadioButton	单选按钮(多选一)，也可分组(每组多选一)
	ListBox	下拉列表框，选项以列表形式显示
	ComboBox	组合框，可以输入的列表框
	ScrollBar	滚动条
	GroupBox	组框，用于组件的分组
	RadioGroup	用于设计单选按钮组
	Panel	面板，可以包含其他组件的容器
	ActionList	系统动作列表

表 1-2 Additional 组件选项卡中最常见的组件

组件图标	组 件 名 称	说 明
	BitBtn	用于创建带位图的按钮
	SpeedButton	加速按钮，可以用于创建工具栏
	Image	图像按钮，用于显示图片
	Splitter	分隔条，用于创建可以改变大小的区域

表 1-3 Win32 组件选项卡中最常见的组件

组件图标	组 件 名 称	说 明
	PageControl	选项卡组件，可以设计多个页面
	ImageList	图像列表，为其他组件提供图标
	RichEdit	多格式编辑器
	StatusBar	状态栏
	ToolBar	工具栏
	CoolBar	设计可改变大小、移动的工具栏，它是容器

1.2 Delphi 7 集成开发环境

Delphi 7 的集成开发环境 IDE(Integrated Development Environment)是设计程序界面、编写程序代码、调试程序、生成可执行文件的软件环境。Delphi 7 的集成开发环境使得开发 Delphi 应用程序变得非常方便。

1.2.1 Delphi 7 集成开发环境组成

Delphi 7 集成开发环境由 6 个部分构成：主窗口、窗体设计器、对象监视器、对象查看树、代码编辑器和代码浏览器，如图 1-2 所示。

集成开发环境的各个部分是一个有机体，它们协同工作。

1. 主窗口

Delphi 7 主窗口是 Delphi 7 集成开发环境的控制中心，它具有 Windows 应用程序的风格。Delphi 7 主窗口包括 3 个部分：主菜单、工具栏和组件面板。

(1) 主菜单

主菜单提供了所有进行程序开发所需要的命令和功能，包括创建、打开、保存各种类型的文件，编辑程序，视图查看，修改选项等功能。Delphi 7 共有 11 个主菜单项，如表 1-4 所示。

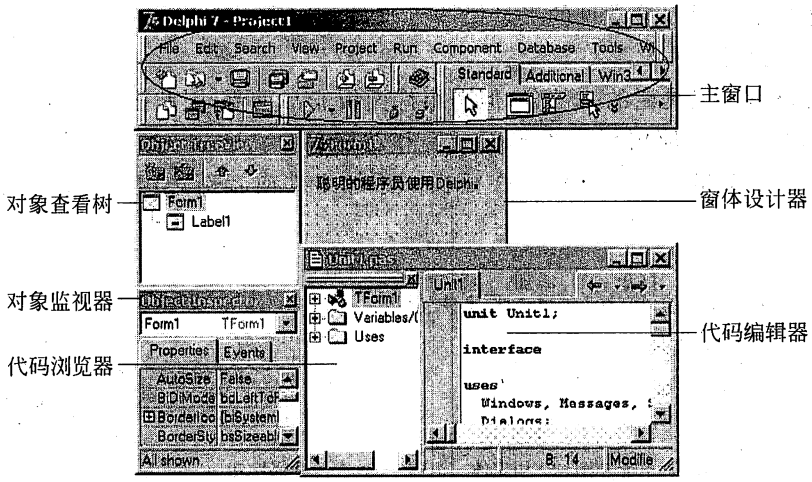


图 1-2　Delphi 7 集成开发环境

表 1-4　Delphi 的主菜单项

主菜单名称	主 要 功 能
File(文件)	新建、打开、保存各种文件
Edit(编辑)	编辑项目中的各种文件
Search(查找)	实现文本的查找
View(浏览)	打开项目管理器、对象查看器，显示调试信息，查看窗体、单元，设置工具栏
Project(项目)	管理项目文件
Run(运行)	提供运行、调试、中断等各种与运行有关的命令
Component(组件)	安装第三方组件，设置组件面板
Database(数据库)	提供与数据库有关的各种工具
Tools(工具)	提供 Delphi 开发环境的设置
Windows(窗口)	切换 Delphi 中的窗口
Help(帮助)	提供 Delphi 帮助信息

(2) 工具栏

Delphi 工具栏(ToolBar)上的每个按钮都可以在菜单中找到，把菜单中经常使用的菜单项做成工具栏的形式，可以使用户操作更加方便，用户无须在菜单中查找，只需直接单击工具栏按钮即可。Delphi 7 提供的工具栏有 7 个，分别是 Standard、View、Debug、Custom、Internet、Desktop 和 Component Palette。如图 1-3 所示为 Delphi 7 的工具栏。

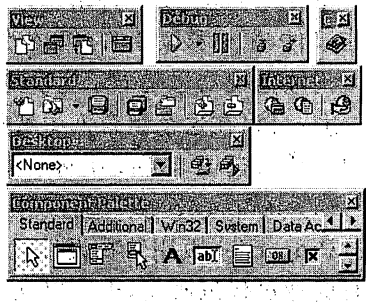


图 1-3　Delphi 7 的工具栏

上面这些工具栏中最常见的工具栏按钮如图 1-4 所示。

(3) 组件面板

组件面板属于工具栏的一部分,但是组件面板非常重要,因此把组件面板单独拿出来讲解。组件是 Delphi 应用程序的界面元素和功能单元。Delphi 7 的组件面板按功能分为 32 页,近 400 个组件。用户可以在面板中添加组件或者新增组件页面。组件面板如图 1-5 所示。

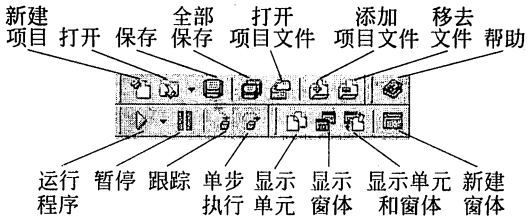


图 1-4 最常用的工具栏按钮

图 1-5 组件面板

组件面板左边的抓柄用于拖动组件面板,用户可以根据自己的习惯来改变组件面板的位置,定制自己喜欢的 IDE 结构布局。对象选择按钮用于切换所选择的组件。选项卡中存放着相同类型的组件,便于用户查找、选择。导航按钮的作用是将未显示出来的选项卡显示出来,供用户选择之用。

2. 窗体设计器

Delphi 窗体设计器是设计应用程序界面的工作区域,运行时程序界面完全与设计时的界面一样。系统自动产生用户界面代码,用户只需要完成相应功能的程序段。

窗体在运行时称为窗口,一般来说一个应用程序不止一个窗口。窗口由标题栏、工作区构成。标题栏给出窗体的标题,右边还有控制按钮,用于在运行期间关闭、最大化、最小化窗口。工作区是添加组件的位置,工作区上有栅格(Grids),用于组件对齐使用,栅格的密度可以调节,在运行期间栅格是不可见的。

3. 对象监视器(Object Inspector)

利用对象监视器可以设置窗体中各个组件以及窗体的属性,或者使各个组件或窗体响应不同的事件。选择菜单 View|Object Inspector 命令可以显示对象监视器。

属性(Property)是对象的一些数据,用于描述对象的颜色、大小、字体等。

事件(Event)是一种消息处理机制,它能够捕捉某种动作并做出一些响应。例如鼠标单击事件、鼠标移动事件、窗体改变大小事件等。

对象监视器总是显示当前选中的组件或者窗体。对象监视器的上面是一个对象组合框,用户可以在这个组合框中选择不同的对象。对象监视器是多页结构,包括属性页和事件页。单击选项卡可以在两个页面之间进行切换,如图 1-6 所示。

(1) Properties 页(属性页)

Properties 页用于在设计时查看或者修改对象的相关属性。对象的属性可以在程序运行时通过代码改变,也可以在 Properties 页中改变,也就是程序运行前设置初始值。根据属性类型的不同,可以采用不同的属性设置方法,如下给出了属性设置的不同方法。

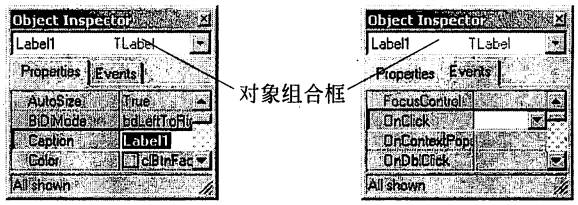


图 1-6　对象监视器

- 属性为数值类型或者字符串类型。只需要直接输入属性值即可，例如 Name、Caption 属性就可以在属性右边的编辑框中直接输入属性值。
- 属性为布尔型或者枚举型。选择该属性，在该属性右边的组合框中选择适当的属性值即可，例如 Label 的 Visible 属性、BitBtn 的 Kind 属性就属于这种类型。
- 通过对话框设置属性。有些属性的右边有按钮，单击该按钮出现对话框，在对话框中可以设置该对象的属性值，例如 Font、Picture、Items、Glyph 等属性就是通过这种方法设置的。
- 直接调节对象的大小和位置也可以设置对象的部分属性，例如 Left、Top、Width 和 Height 等。
- 子属性设置。有的属性还有子属性。例如 Font，它的左边有一个⊞符号，单击该符号则⊞变成⊟号，同时 Font 的子属性被展开，此时用户可以设置其子属性 Color、Name、Size 等。
- 属性为集合类型。单击该属性左边的⊞符号，该⊞号变成⊟号，然后再设置该属性的集合元素。例如 Anchors 就属于这种类型。

(2) Events 页(事件页)

在 Events 页面中可以设置对象或者窗体的响应事件。不同对象预置了不同的事件，例如 OnClick、OnActivate、OnKeyPress 等。在编写程序的时候要为某些预置的事件编写响应的代码程序。

【例 1-1】 编制一个简单的 Windows 应用程序。

说明：本程序保存在“d:\delphi 程序\ch1\eg1-1”文件夹中，以后的程序也保存在相应的文件夹中。请读者在编写 Delphi 程序的时候养成新建文件夹的习惯，本书中，编写程序前新建文件夹这一步骤一律省略。

步骤如下：

(1) 在窗体上添加按钮 Button1 和标签 Label1，调节好窗体 Form1、Button1 和 Label1 的大小和位置。

(2) 设置 Button1 的 Caption 属性为“确定”，界面如图 1-7 所示。

(3) 编写代码，选中 Button1，此时在 Events 中显示的是 Button1 对象，在对象监视器中选中 Events 页，并找到 OnClick 事件，用鼠标双击右边空白栏，如图 1-8 所示。此时打开代码编辑器，系统自动在窗体 Form1 的单元 Unit1 的 Interface(接口)部分插入该过程的声明，同时在 Implementation(实现)部分插入该过程的框架，如图 1-9 所示。

(4) 在事件过程框架中也就是在 begin 和 end 之间(光标所在处)输入代码：

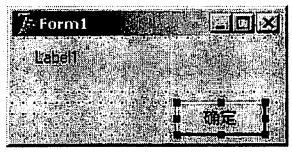


图 1-7 编写 Windows 应用程序

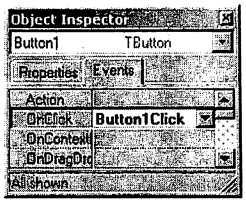


图 1-8 双击 Events 页的 OnClick 右边空白栏

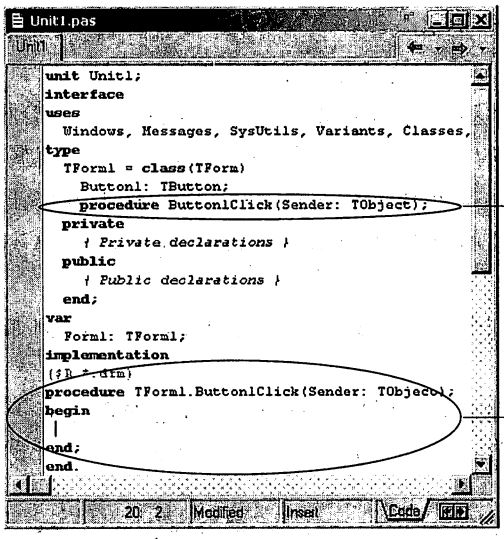


图 1-9 在代码编辑器中输入事件过程

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 label1.Caption:='聪明的程序员使用 Delphi!';          //本行为输入的代码
end;
```

(5) 按 F9 键运行,结果如图 1-10 所示。

4. 对象查看树(Object TreeView)

对象查看树以树状结构的形式显示窗体中各个组件之间的逻辑关系。对象查看树和对象监视器是同步协调工作的。在设置属性或者编写事件过程的时候,可以在对象查看树中选择好某个对象,然后再在对象监视器中设置属性或者编写事件过程。这样远比直接在对象监视器的对象组合框中选择对象方便。对象查看树如图 1-11 所示。

图 1-10 程序运行结果

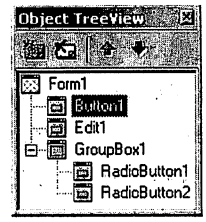


图 1-11 对象查看树

如果对象查看树被关闭，选择菜单 View|Object TreeView 可以显示对象查看树。

5. 代码编辑器

代码编辑器是程序代码输入和编辑的工具。尽管可视化编程技术自动生成了一些代码，但是用户仍然有很多代码需要手工完成，因此代码编写仍然是程序设计的核心工作。Delphi 的代码编辑器是一个功能强大、使用方便的代码编写工具。

(1) 代码编辑器的组成

代码编辑器是一个多页的文本编辑器。有时候应用程序由几个窗体构成，此时单元就不止一个。通过选项卡可以选择不同的页面，每个页面就是一个单元文件。代码编辑器的标题显示单元文件名。如图 1-12 所示为代码编辑器。

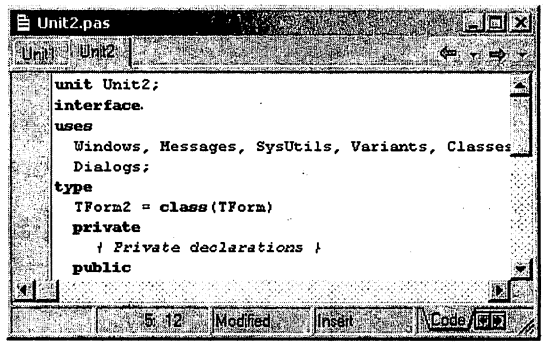


图 1-12　代码编辑器

(2) 代码编辑器的功能

Delphi 的代码编辑器具有友好的提示和帮助功能，它主要表现在如下几点：

① 代码完善功能。在用户输入已创建的对象的名称和点号的时候，稍作停顿，系统将自动弹出提示列表框，列出该对象的所有属性和方法。用户无须自己输入，只需在列表中选择合适的属性和方法，则该属性或者方法就自动添加程序行中，如图 1-13 所示。

② 参数提示功能。在输入代码的时候，用户输入对象的方法或者标准的自定义的函数或者过程的时候，当用户输入左括弧，稍作停顿，系统会给出包括参数个数、参数类型在内的参数提示。参数提示功能如图 1-14 所示。

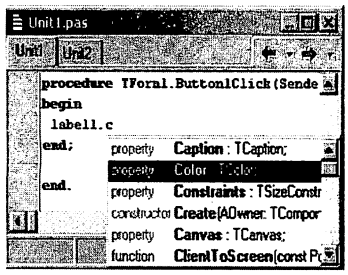


图 1-13　代码编辑器的代码完善功能

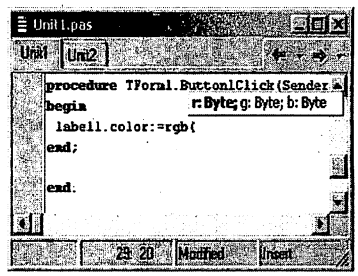


图 1-14　代码编辑器的参数提示功能

③ 代码模板功能。代码模板用于提供语句模板以帮助代码编写。例如，用户输入 for，然后按 Ctrl+J 键，系统自动显示 for 循环语句的语法供用户参考，防止代码出错。如图 1-15 所示为代码模板功能。左图列表框给出 for 语句的两种格式，第一种是带 begin 和 end 的格式，第二种是不带 begin 和 end 的格式；右图是选择第一种 for 循环格式，并按回车键后显示的 for 语句格式。

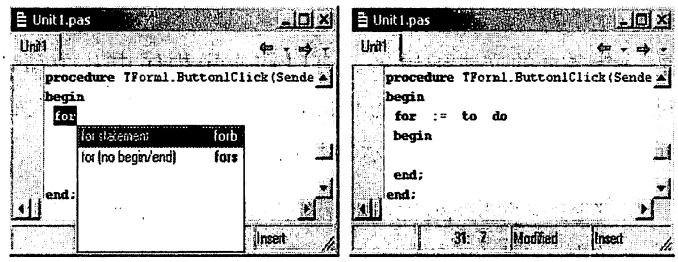


图 1-15 代码模板功能

④ 符号洞察功能。将鼠标置于 Delphi 标识符（函数名、过程名、组件名、类型名和变量名）上，稍作停顿，系统自动给出该标识符的相关信息，如图 1-16 所示。

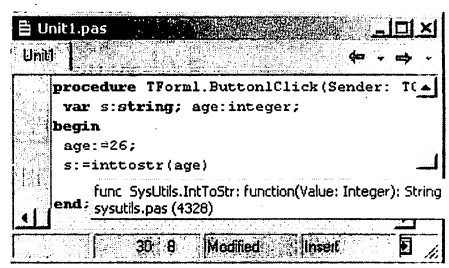


图 1-16 代码编辑器的符号洞察功能

6. 代码浏览器

代码浏览器以树形视图方式显示代码编辑器中的单元文件。通过代码浏览器可以非常方便地在各个单元之间漫游或者在某个单元中加入新的元素。代码浏览器和代码编辑器统一协调工作。

1.2.2 Delphi 应用程序所包含的文件

一个 Delphi 应用程序由多个文件构成。表 1-5 给出了 Delphi 中一些常见的文件类型及含义。

表 1-5 Delphi 应用程序所包含的文件

文件类型	扩展名	说明
项目组文件	bgp	
包文件	dpk	
项目文件	dpr	记录应用程序所包含的单元
单元文件	pas	Object Pascal 源代码文件
窗体描述文件	dfm	描述窗体和组件的性质
单元编译文件	dcu	单元编译后生成的文件
选项文件	dof	记录项目的选项设置

续表

文件类型	扩展名	说明
资源文件	res	
动态链接库文件	dll	
配置文件	cfg	保存项目的配置
可执行文件	exe	
备份文件	～pas、～dfm、～dpr	单元、窗体、项目文件的备份文件

我们以例 1-1 为例来看看 Delphi 中的一些主要文件。上次编写例 1-1 程序保存文件时，系统只会自动保存项目文件和单元文件，而其他一些文件是在编译、运行、生成“.exe”文件过程中形成的。先打开例 1-1 中的项目文件和单元文件，并编译该项目和生成 project1 .exe 文件。此时系统形成如下几个文件：project1. cfg、project1. dof、project1. dpr、project1. exe、project1. res、unit1. dcu、unit1. dfm 和 unit1. pas。

1. 单元文件

单元是扩展名为 pas 的源程序代码，单元中可以定义或者声明常量、变量、函数、过程、对象、组件等。设计单元的好处是，单元可供其他应用程序共享。单元还可以独立编译为扩展名为 dcu 的文件，编译后的 dcu 文件可以连接到可执行文件中。单元文件不能够独立运行。单元文件分为如下几种。

(1) 与窗体相关的单元文件，这种单元文件与窗体一一对应。

(2) 用于存储公用函数、过程、常量、变量的单元文件，例如 math。

(3) 组件文件，该单元提供组件开发接口。

单元文件是有一定结构的，一般由名称、接口、实现和初始化 4 个部分组成。各个部分说明如下。

(1) unit 子句。标识单元名，单元名与单元文件的文件名相匹配。

(2) interface 接口部分。说明本单元哪些部分可以供其他单元或者程序访问，在此声明的函数、过程、类型、常量、变量可供其他单元或者程序共享。接口部分只有声明，不能有定义，定义在实现部分完成。

(3) implementation 实现部分。接口部分声明的函数、过程、类型、常量、变量等的具体实现在此完成，实现部分也可以说明函数、过程、类型、常量、变量等，但是它们仅供本单元使用。

(4) initialization 初始化部分。可选部分，用于初始化程序所需数据。

请看例 1-1 中的单元文件 project1. pas。

```
unit Unit1;                                    //说明单元

interface                                      //接口部分

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
```

```
  Dialogs, StdCtrls,math;

type                                                    //类型说明
  TForm1=class(TForm)
    Label1: TLabel;
    Button1: TButton;
    procedure Button1Click(Sender: TObject);            //过程声明
  private
    {Private declarations}                              //私有部分
  public
    {Public declarations}                               //公有部分
  end;

var                                                     //变量说明
  Form1: TForm1;

Implementation                                          //实现部分

{$R*.dfm}

procedure TForm1.Button1Click(Sender: TObject);         //具体实现上面声明过的过程
begin
  label1.Caption:='聪明的程序员使用 Delphi!';
end;

end.                                                    //end之后有点号,是程序结束标志
```

说明：在接口部分的 uses 短语后有一些单元名称，例如 Windows、Messages、SysUtils 等，这些单元是系统的标准单元或者是其他程序的单元，本单元要使用它们，因此需要在接口部分的 uses 短语后指明这些单元。

2. 窗体描述文件

每个窗体都包含一个扩展名为 dfm 的窗体描述文件，该窗体描述文件用于描述窗体及窗体内组件的属性。鼠标右键单击窗体选择 View as text 菜单可以显示窗体描述文件。请看例 1-1 中的窗体描述文件 Unit1.dfm。

```
object Form1: TForm1
  Left=192
  Top=107
  Width=205
  Height=104
  Caption='Form1'
  Color=clBtnFace
  Font.Charset=DEFAULT_CHARSET
  Font.Color=clWindowText
  Font.Height=-11
  Font.Name='MS Sans Serif'
  Font.Style=[]
  OldCreateOrder=False
```

```
  PixelsPerInch=96
  TextHeight=13
  object Label1: TLabel
    Left=16
    Top=8
    Width=153
    Height=25
    AutoSize=False
    Caption='Label1'
  end
  object Button1: TButton
    Left=120
    Top=48
    Width=65
    Height=25
    Caption='确定'
    TabOrder=0
    OnClick=Button1Click
  end
end;
```

细心的读者一定可以从窗体文件中看到窗体及窗体内组件的一些属性。窗体描述文件就是在设计应用程序界面时系统自动生成的一些代码。鼠标右键单击窗体描述文件选择 View as form 可以重新以界面的形式显示窗体。

3. 项目文件

项目文件的扩展名为 dpr。项目文件是应用程序的宏观框架，它管理着应用程序的多个单元。

以例 1-1 为例；请看项目文件 project1. dpr。选择菜单 View|Units...，出现 View Unit 对话框，在对话框中选择项目文件 Project1，按 OK 按钮。View Unit 对话框如图 1-17 所示。

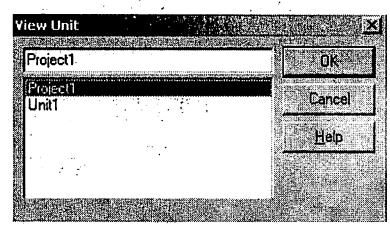


图 1-17　View Unit 对话框

project1. dpr 文件如下：

```
program Project1;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};
{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

1.2.3 组件的画法

为了创建 Delphi 应用程序，需要在窗体上添加各种组件，并且调整这些组件的位置和大小。在窗体上画组件，包括组件的添加、组件的选定、改变组件的大小和位置、组件的复制和删除、组件的对齐等。

1. 添加组件

将组件添加到窗体上，有如下几种方法。

(1) 单击所需组件，然后在窗体适当位置上拖动鼠标画出组件，即可完成组件的添加。

(2) 双击所需组件，即可添加该组件到窗体的中心位置，然后用鼠标调整该组件的大小和位置。

(3) 按住 Shift 键不放，用鼠标单击所需组件，此时该组件出现蓝色边框，同时对象选择按钮弹起，用鼠标在窗体适当位置拖动即可连续添加该组件。用鼠标单击组件选择按钮可以取消该组件的连续添加操作。

2. 选定组件

为了操作组件必须首先选定组件，被选定的组件边框四周有 8 个黑色的小方块，表明该组件是活动组件，也称之为当前组件。鼠标单击组件可以选定该组件。还可以同时选定多个组件，方法有 3 种。

(1) 按住 Shift 键不放，再用鼠标单击所要选定的各个组件，则鼠标单击的各个组件都被选中。

(2) 按住鼠标左键在窗体上画一个矩形区域，与这个矩形区域相交的组件都被选中。

(3) 利用鼠标、Shift 键、Ctrl 键在对象监视器中也可以选定多个组件。

选中的多个组件四角有灰色小方块。多个组件的选中如图 1-18 所示。

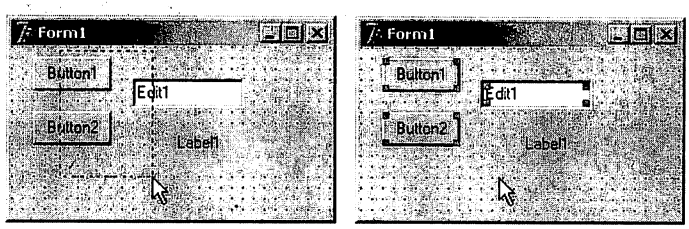


图 1-18 选定多个组件

3. 组件大小和位置的改变

首先选定组件(一个或者多个)，再按住 Shift 键，最后通过按方向键可以改变组件的大小。选定组件(一个或者多个)，再按住 Ctrl 键，最后通过按方向键可以改变组件的位置。这些操作也可以在对象监视器中通过设置属性 Left、Top、Width 和 Height 来实现。

4. 组件的复制和删除

在组件的操作中，为了更加快速操作组件，经常要复制或者删除组件。先将所要操作

的组件选定，然后按 Ctrl＋C 键可以将选定的组件复制到剪切板，而按 Ctrl＋V 键可以将此前剪切的组件粘贴到窗体。对于活动组件只需要按 Del 键就可以删除该组件。

上面这些操作也可以通过 Edit 菜单来实现。

5. 组件的对齐

当窗体上有多个组件的时候，如果对齐这些组件，界面会显得更加美观、整洁。Delphi 提供两种对齐工具用于水平方向或者垂直方向的对齐。这两种工具是对齐对话框和对齐面板。选择菜单 Edit | Align... 可以打开对齐对话框，选择菜单 View | Alignment Palette 则可以打开组件对齐面板。如图 1-19 所示，左边为对齐对话框，右边为对齐面板。

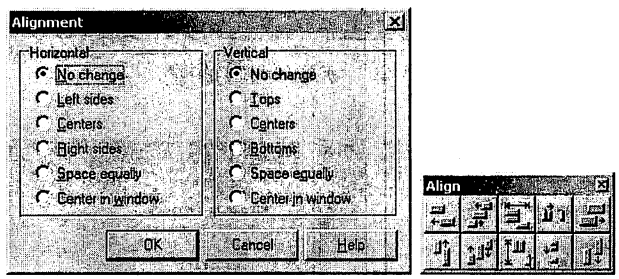


图 1-19　对齐对话框和对齐面板

对齐面板和对齐对话框是一一对应的关系，表 1-6 给出了它们的功能。

表 1-6　对齐对话框和对齐面板功能说明

上排按钮	左边单选项	说　明	下排按钮	右边单选项	说　明
	No change	水平方向不改变		No change	垂直方向不改变
	Left sides	选定组件左边对齐		Tops	顶边对齐
	Centers	水平居中对齐		Centers	垂直居中对齐
	Center in window	水平居中		Center in window	垂直居中
	Space equally	水平间距相等		Space equally	垂直间距相等
	Right sides	选定组件右边对齐		Bottoms	底边对齐

1.2.4　创建控制台应用程序

Delphi 应用程序包括控制台应用程序和 Windows 应用程序。前者是基于 DOS 环境的，在 Windows 之前的程序都是基于 DOS 环境的。虽然现在大多数情况下使用基于 Windows 环境的应用程序，但是有时候也使用基于 DOS 环境下的控制台程序。

下面创建一个控制台应用程序。

【例 1-2】 创建控制台应用程序，输出“Hello，World!”。

步骤如下：

(1) 选择 Delphi 菜单 File | New | Other...，出现新建项目对话框，如图 1-20 所示。

在对话框中选择 Console Application，出现代码编辑窗口，如图 1-21 所示。

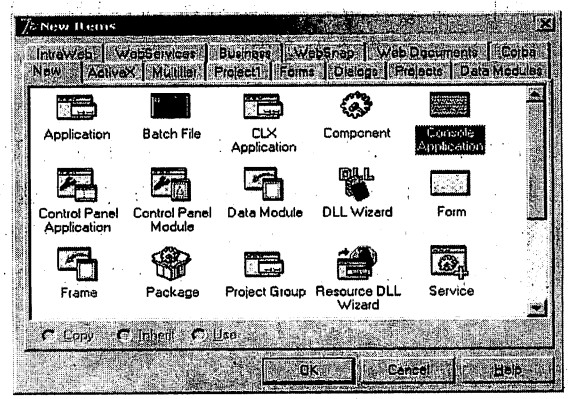


图 1-20　新建项目对话框

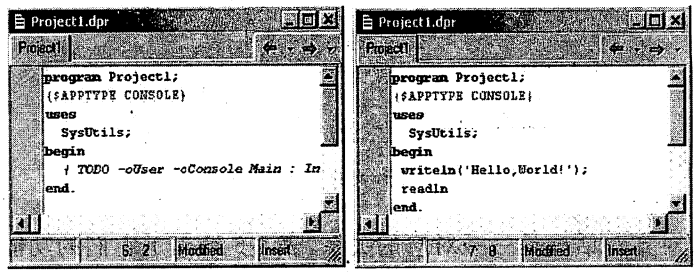


图 1-21　代码编辑窗口(右图输入了代码)

(2) 在代码编辑窗口的 begin 和 end 之间输入代码，如下：

```
program Project1;

{$APPTYPE CONSOLE}
uses
  SysUtils;

begin
  writeln('Hello,World!');                    //输入的代码,用于显示"Hello,World!"
  readln;                                     //输入的代码,用于等待按回车键
end.
```

(3) 运行程序，结果如图 1-22 所示。

(4) 如果对程序运行结果满意，可以保存程序。一般来说，一个程序最好保存到一个单独的文件夹中，这样便于管理。单击工具栏的按钮(保存全部)，保存文件到文件夹"d:\delphi 程序\ch1\eg1-2"，工程文件名为 project1. dpr。此时在文件夹"d:\程序"中自动生成其他文件，如 Project1. cfg 和 Project1. dof。

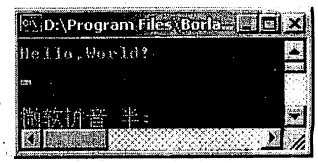


图 1-22　程序运行结果

(5) 生成可执行文件。如果需要，可以将程序编译成可执行文件，选择 Project|Build Project1 将程序编译为可执行文件。

1.2.5　创建 Windows 应用程序

Delphi 应用程序包括控制台应用程序和 Windows 应用程序，其中 Windows 应用程序使用非常广泛。与控制台应用程序相比，Windows 应用程序需要使用窗体、组件等界面元素，因此 Windows 应用程序需要设计界面。与面向过程的程序设计相比，Delphi 面向对象的应用程序采用事件驱动，系统完成事件过程框架，其余的程序代码还需要用户自己完成。Delphi 的 Windows 编程步骤一般分为：

(1) 设计界面。利用组件设计应用程序的界面，在窗体上添加组件，调节组件的大小和位置到合适为止。程序运行时的界面就是设计时的界面。

(2) 设置属性。每个组件都有自己的一些属性，有的组件和窗体在程序运行之前需要设置其属性。

(3) 编写代码。界面设计、属性设置完成之后，大部分工作是编写代码。

(4) 运行程序、保存程序。代码设计完成之后要运行程序，如果满意可以保存程序，甚至还可以生成可执行文件或者安装程序。

1. 创建工程

在 Delphi 中，每个应用程序都称为一个工程。编写一个应用程序是从创建一个项目开始的。Delphi 中的项目有很多种，本节介绍如何创建一个 Windows 应用程序的项目。

选择 File|New|Application 可以创建一个 Windows 应用程序。工程文件默认为 Project1、Project2 等，单元文件默认为 Unit1、Unit2 等。

2. 添加组件

窗体和组件是 Delphi 应用程序的界面，也是 Delphi 应用程序的功能部件。现在在窗体上添加按钮 Button1 和 Button2、标签 Label1，调节组件和窗体的大小、位置，如图 1-23 所示。

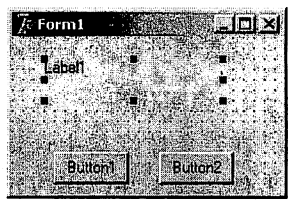


图 1-23　程序界面

3. 设置属性

选中窗体 Form1，在对象监视器中设置窗体 Form1 的 Caption 为“Hello，World!”。

选中 Button1，在对象监视器中设置 Button1 的 Caption 为“显示”；选中 Button2，在对象监视器中设置 Button2 的 Caption 为“关闭”。

选中 Label1，在对象监视器中选择 Color 属性，在右边的下拉组合框中选择颜色 clRed，如图 1-24 所示。设置 Label1 的 Visible 属性为 False。用鼠标单击 Font 属性右边的按钮，此时出现字体对话框，在字体对话框中设置 Label1 的 Font 属性，如图 1-25 所示。

各个对象的属性设置见表 1-7。

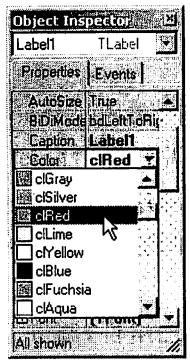


图 1-24 设置颜色属性

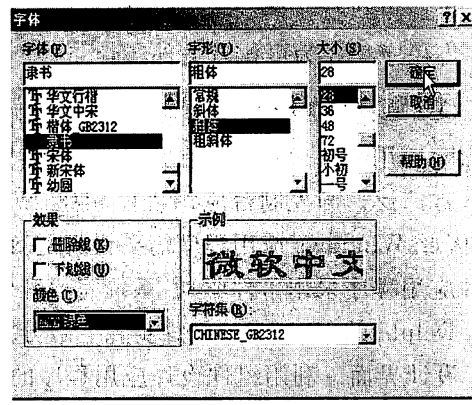


图 1-25 设置字体属性

表 1-7 对象的属性设置一览表

对 象	属 性	属 性 值	说 明
Form1	Caption	Hello,World!	窗体标题
Label1	Color	clRed	背景为红色
	Font. Color	clGreen	字体颜色为绿色
	Font. Size	28	字体大小
	Font. Name	隶书	字体名称
	Font. Style	[fsBold]	字体形状：粗体
Button1	Caption	显示	按钮标题
Button2	Caption	关闭	按钮标题

设置属性后的界面如图 1-26 所示。

4. 编写代码

在对象监视器中选定 Events 选项卡，并在对象监视器的对象组合框中选择 Button1。在事件选项卡中选择 OnClick 事件，鼠标右键单击空白栏，此时出现代码编辑器，如图 1-27所示。在代码编辑器中发现事件过程的框架已经生成，用户只需要添加相应的代码即可。

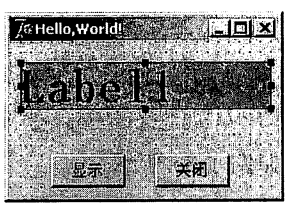


图 1-26 设置属性后的界面

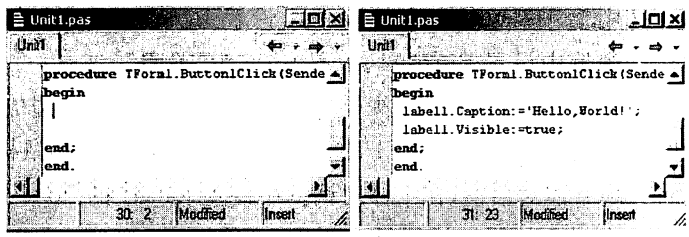


图 1-27 代码编辑器(右边已加入了代码)

在 begin 和 end 之间输入代码：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 label1.Caption:='Hello,World!';
 label1.Visible:=true;
end;
```

用同样的方法，为 Button2 添加 OnClick 事件过程。代码如下：

```
procedure TForm1.Button2Click(Sender: TObject);
begin
 close
end;
```

5. 运行程序

按 F9 键运行程序，单击“显示”按钮显示“Hello，World!”，单击“关闭”按钮程序停止，如图 1-28 所示。

图 1-28　程序运行结果

6. 保存程序

程序设计好之后，如果运行正常，用户感觉满意，可以先保存起来，以免丢失。首先为上面的程序建立一个文件夹“d:\程序 2”，然后选择菜单 File|Save All 或者选择工具栏按钮分别保存单元文件和项目文件。如果程序被保存过，则不会出现保存对话框，而是直接以原来的文件名再次保存。

其他与保存有关的菜单见表 1-8。

表 1-8　保存菜单命令及说明

菜 单 命 令	说　　明
File|Save	保存单元文件，如果是首次保存则同 Save As
File|Save As	打开保存单元文件对话框，将当前单元文件更名保存
File|Save Project As	打开保存工程文件对话框，将当前工程文件更名保存

1.3　小结

本章介绍了 Delphi 7 的特点和优点、对象的一些基本概念，介绍了类和组件的基本概念，还介绍了 Delphi 7 集成开发环境。读者应该熟练掌握 Delphi 7 集成开发环境。

习题

1. Delphi 有哪些特点和优点？
2. 什么是对象的属性、方法和事件？

3. 什么是事件驱动？什么是事件过程？
4. Delphi 集成开发环境是由哪几部分构成的？
5. 怎样修改对象的属性？
6. 如何建立控制台应用程序？请读者自己设计一个控制台应用程序。
7. 如何建立 Windows 应用程序？请读者自己设计一个 Windows 应用程序。
8. Delphi 最常见的文件有哪些种类？分别表示什么含义？

第 2 章

Object Pascal 语言

Delphi 是基于 Object Pascal 语言的 Windows 应用程序开发软件，Object Pascal 语言是 Pascal 语言的扩展，它支持类的封装、继承、多态等面向对象的编程技术。Object Pascal 语言具有丰富的数据类型和流程控制，语法简单、易学，是学好 Delphi 的基础。

2.1 语言成分

本节将介绍 Pascal 语言的程序组成，包括程序结构、关键字、标识符、分隔符、注释等。

1. 程序结构

下面是 Object Pascal 语言的程序结构，它包括程序名称、说明部分（包括函数、过程、类型、变量、常量的说明或定义）和程序体。

```
program 程序名;

单元引用;
常量说明;
类型定义;
变量定义;
过程与函数的定义;
begin
  语句 1;
   ⋮
  语句 n;
end.
```

说明：

（1）程序首部以关键字 program 开始，其后是程序名称，程序名称是 Pascal 标识符。

（2）说明部分说明了本程序使用到哪些基本的单元，还包括常量的说明、类型的定义、变量的定义、过程与函数的定义等。

（3）程序体是以 begin 开始的 end 结束的语句序列，每个语句最后都有一个分号，end 之前的分号可以省略，程序最后以点号为结束标志。

2. 关键字和标识符

关键字(Key Word)又叫做保留字(Reserved Word),是系统事先定义好的有特殊含义并规定不得重新定义的词汇。Object Pascal 定义了 65 个关键字,参见表 2-1。

表 2-1 Object Pascal 关键字

and	div	function	library	property	to
array	do	goto	mod	raise	try
as	downto	if	nil	record	type
asm	else	implementation	not	repeat	unit
begin	end	in	object	resourcestring	until
case	except	inherited	of	set	uses
class	exports	initialization	or	shl	var
const	file	inline	out	shr	while
constructor	finalization	interface	packed	string	with
destructor	finally	is	procedure	then	xor
dispinterface	for	label	program	threadvar	

说明:除了上面这些保留字外,private、public、protected、automated 在定义对象类型的时候也作关键字使用,而在其他地方这几个词汇表示一般的标识符。

标识符是用来表示变量、常量、类型、函数、过程、对象等名称的符号,Pascal 标识符分为标准标识符和自定义标识符。

所谓标准标识符,就是系统预先分配给一些标准函数、过程、类型以及标准文件使用的符号。例如:

(1) 标准常量。如 False、True 等。

(2) 标准类型。如 Integer、Boolean、Real 等。

(3) 标准函数。如 sin、cos、abs 等。

(4) 标准过程。如 get、put 等。

(5) 标准文件。如 Input、Output 等。

所谓自定义标识符,就是程序员根据程序需要为自己定义的常量、变量、函数、过程以及其他一些名称。Object Pascal 规定标识符应该遵循如下命名规则:

(1) 标识符由大小写英文字母、阿拉伯数字、下划线构成,长度不超过 255。

(2) 必须以英文字母或者下划线开头。

(3) 不得与关键字同名。

(4) 最好不要与标准标识符同名,以免混淆。

说明:

(1) 标识符最好做到见名知义,例如用 age 表示年龄、stu_name 表示学生姓名等。

(2) 长度不要太长,否则容易出错,例如 studentname 可以写成 stu_name 等。

(3) Pascal 标识符不分大小写。

3. 分隔符和注释

分隔符用于分隔两个语法单位，Object Pascal 语言的分隔符分为两种：

(1) 分号。用于分隔语句。

(2) 逗号。用于分隔数据。

注释是程序中用于解释或者说明的部分，在程序运行中注释不被执行。程序注释的作用有两个：

(1) 调试程序的作用。在调试程序的过程中，如果发现某些语句不需要或者有错误，用户不要急于删除这些语句，可以将这些语句加上注释，等到调试通过再来处理这些语句。这样可以避免误删。

(2) 注释作用。对于那些难理解的语句，可以适当加一些注释语句，这样可以增强程序的可读性。

Pascal 语言提供两种注释：

(1) 单行注释。注释号“//”之后的内容就是注释。

(2) 多行注释。注释号“{”和“}”之间就是注释，也可以用“(*”和“*)”来注释多行。

2.2 数据类型

描述客观事物的数字、字符以及所有能够输入到计算机中并能被计算机接受的各种符号的集合统称为数据(data)。数据是计算机程序处理的对象，数据的一个重要特征就是这个数据属于什么类型。数据类型不仅规定了该类型的数据的表示形式、取值范围，还规定了这种数据类型所能参加的各种运算。

Object Pascal 继承了 Pascal 数据类型丰富的特点，提供了丰富的数据类型，参见表 2-2。

表 2-2　Object Pascal 数据类型

类　型	名　称	类　型	名　称
整型	Integer	记录类型	Record
实型	Real	文件类型	File
字符型	Char	类类型	Class
字符串型	String	类引用类型	Class Reference
布尔型	Boolean	接口类型	Interface
枚举型	Enumerated	指针类型	Pointer
子界型	Subrange	过程类型	Procedural
集合类型	Set	可变类型	Variant
数组类型	Array		

说明：

(1) 整型、实型、字符型、字符串型、布尔型这 5 个类型可以直接使用，称为标准类型，而枚举、子界等共 12 个其他类型需要先定义再使用。

(2) 整型、字符型、布尔型、枚举型和子界型称为顺序类型。关于顺序类型，将在后面详细介绍。

2.2.1 数值型数据

Object Pascal 数值型数据类型又分为整型和实型。

1. 整型

整型的数据类型有正整型、零、负整型，含义与数学中的含义相同。整型数据类型是顺序类型。例如，1 的前导是 0，1 的后继是 2。

整型又分为很多类型，如表 2-3 所示。

表 2-3 整型数据类型

分 类	类 型	字节数	取值范围
有符号整型	Shortint	1	−128～127
	Smallint	2	−32768～32767
	Integer	4	-2^{31}～$2^{31}-1$
	Int64	8	-2^{63}～$2^{63}-1$
无符号整型	Byte	1	0～255
	Word	2	0～65535
	Longword	4	0～$2^{32}-1$

2. 实型

实型的数据有整数部分和小数部分，含义和数学中一样。实型数(简称实数)不是顺序类型。实数有两种表示方法：小数表示方法和科学计数表示方法。例如，12.3450 就是小数表示形式，而 1.3570000E+2 就是科学计数表示形式。E 之前的数字称为尾数，表示精度；E 之后的数表示阶码，表示 10 的次方数，阶码必须是整数。

下面都是合法的实数：

1.0　　123.0　　0.0　　−0.0　　0.8

1E−3　　1.6E−7　　3.5E4

下面的实数表示方法有错误：

12.　　.234　　E6　　2E0.6　　12

说明：

(1) 实数的小数表示方法规定，小数点两边都必须有数字。

(2) 科学计数表示方法规定，E 的前后都必须有数字，之后的数为整数。

实型数如表 2-4 所示。

表 2-4　实型数

类　型	说　明	取值范围
Single	4 字节浮点数，精度 8 位	1.5E－45～3.4E＋38
Real，Double	8 字节浮点数，精度 16 位	5.0E－324～1.7E＋308

2.2.2　字符型数据

Object Pascal 字符型数据类型又分为字符型和字符串型。

1. 字符型

字符型(char)也是顺序类型，表示单一的字符，如'A'，'a'，''(不是")，'#'，'*'等。在计算机内存里字符以它的 ASCII 码的形式保存。因此字符可以用单引号的形式表示，还可以用#加 ASCII 码的形式表示，如#65 表示'A'，而#97 表示'a'。#常用来表示一些不显示的字符，如#13 表示回车，#8 表示 BackSpace，#32 表示空格，#0 表示空字符，#27 表示 Esc 等。

注意：两个单引号不表示空字符。

除了 char 外，还有一种字符型 widechar，表示一个 16 位的 Unicode 字符。

2. 字符串型

由多个字符组成的字符序列就是字符串。例如'abc'就是一个字符串。Object Pascal 包含 4 种类型的字符串，如表 2-5 所示。

表 2-5　字符串型

类　型	名　称	最大长度	类　型	名　称	最大长度
短字符型	shortstring	255 个字符	宽字符型	widestring	2^{30}个字符
长字符型	ansistring	2^{31}个字符	字符串型	string	2^{31}个字符

string 使用最为广泛。Delphi 中 Text 属性、Caption 属性都是 string 类型。

2.2.3　布尔型数据

布尔型数据用于逻辑计算，表示事件的真假，它只有两个值真(True)和假(False)。它是顺序类型，False 的序号是 0，True 的序号是 1。False 的后继是 True，True 的前导是 False。

2.3　常量和变量

有两种数据，一种是在程序运行的过程中值保持不变，称为常量。还有一种数据就是在程序运行过程中值是可以改变的，这种数据称为变量。

2.3.1　常量

常量(constant)分为直接常量和符号常量两种。

1. 直接常量

直接常量就是在程序中可直接引用的常数，如整型的常数、实型的常数、字符型常数、

字符串型常数、布尔型常数等。

(1) 整型常数：如 100,0,－123 等。

(2) 实型常数：如 1.2,0.0,－0.0,1E6,2.8E－4 等。

(3) 字符型常数：如＃8,＃27,＃13,'a'等。

(4) 字符串型常数：如'asd','1＋2'等。

(5) 布尔型常数：如 True 和 False。

2. 符号常量

符号常量是以标识符形式出现的常量。如果在程序中某个常数经常出现，就可以定义一个标识符来代替这些常数，这个标识符就是符号常量。定义符号常量使用常量定义语句，语法格式如下：

```
const
<常量名 1>[:<类型名>]=<表达式 1>;
⋮
<常量名 n>[:<类型名>]=<表达式 n>;
```

说明：

(1) const 是 Object Pascal 标识符，表示常量定义段的开始。

(2) 常量名是合法的 Object Pascal 标识符。

(3) 表达式可以是直接常量或者是由已经定义过的符号常量组成的表达式。

(4) 可以在定义常量的时候，定义常量的类型。

【例 2-1】 请看下面的常量定义。

```
const
  pi=3.1415926;                    //常量定义
  alph=pi/3;                       //pi 此前已经定义过,可以直接使用
  pi=3.1416;                       //出错,pi 不可以重新定义
  enterchar:char=#13;              //定义常量并定义类型
  pi2:real=3.1416;
  xingming:string[8]='zhangsan';
```

建议在常量命名的时候，尽量做到见名知义。这样可以提高程序的可读性，使程序更容易懂，更容易修改。

2.3.2 变量

变量(variable)是指在程序执行过程中其值可以改变的量，变量其实就是某个内存地址的标识符。

变量有如下 3 个特征：

(1) 变量名。变量名就是变量的标识符，实际上就是所代表的内存单元的符号地址。

(2) 类型。变量的类型决定了这个数据表示数的范围，占据的内存单元的多少，可以进行哪些操作运算。

(3) 变量的值。变量的值就是所代表的内存单元中存放的数据的值。

在 Object Pascal 中变量要先定义，然后才能使用。变量的定义格式为：

```
var
<变量名 1>: <类型名 1>;
<变量名 2>: <类型名 2>;
⋮
<变量名 n>: <类型名 n>;
```

当多个变量具有相同的类型时，可以使用比较紧凑的格式：

```
var <变量名 1>,<变量名 2>,<变量名 n>: <类型名>;
```

说明：

(1) var 是关键字，表示变量定义段的开始。

(2) 变量名是合法的 Object Pascal 标识符。

(3) 类型名是 Object Pascal 的标准类型或者是其他 Object Pasacl 数据类型。

(4) 同一作用范围的变量名不能同名。

【例 2-2】 请看下面的变量定义。

```
type
  month=1..12;                          //定义类型

var
  x,y,z:real;                           //3 个变量之间用逗号隔开
  i,j,k:integer;                        //定义整型
  ch:char;                              //字符型变量
  m1,m2:month;                          //month 类型已经定义
  days:array[1..12]of integer;          //数组类型
```

变量定义以后就可以在程序中使用了，包括给变量赋值，在表达式中使用变量等。变量定义之后最好要给变量赋一个初始值，因为 Object Pascal 编译器不会自动对变量初始化，如果不初始化变量，变量的值将是一个随机的值，可能会对程序运行造成影响，甚至出错。

2.4　运算符和表达式

运算是程序加工数据的过程，描述程序不同运算的符号称为运算符，参与运算的数据称为操作数。

表达式是表示某个求值规则的运算公式，它包含运算符、操作数、函数、括号等。最简单的表达式是常量、变量或者函数。表达式可以计算出唯一的值，表达式的类型由表达式的值决定，而表达式的值取决于运算符。为此，Object Pascal 提供了丰富的运算符，包括算术运算符、位运算符、字符串运算符、关系运算符、逻辑运算符等。

2.4.1　算术运算符

算术运算符有＋、－、＊、/、div、mod。其中＋、－、＊、/可以完成加、减、乘、除四则运算。而 div 和 mod 分别用于求整除的商和余数，div 和 mod 运算符要求操作数必须是整型数。

【例 2-3】 请看下面算术表达式的值。

```
7 mod -2;          //值为 1
-7 mod 2;          //值为-1
-7 mod -2          //值为-1
7 div -2;          //值为-3
-7 div -2;         //值为 3
-7 div 2;          //值为-3
```

假设 a,b,m,n 都是整数,且 b 不为 0。如果 m 是 a 整除以 b 的商,n 是 a 整除以 b 的余数,则有表达式 a=m * b+n 恒成立。

【例 2-4】 假设今天星期 w,w=0 表示星期天、w=1 表示星期一等,求明天是星期几,昨天又是星期几?

经过分析,明天星期几应表示为:

```
w:=(w+1) mod 7;
```

昨天星期几表示为:

```
w:=(w-1+7) mod 7;
```

说明:w 在 0～6 之间,故(w+1) mod 7 也在 0～6 之间。当 w 的值在 0～5 之间时,(w+1) mod 7 就是 w+1 的值;当 w=6(表示星期六)时,其下一个序号应该为 w=0(表示星期天),而 w=6 时(w+1) mod 7 的值也刚好为 0。综合两种情况,(w+1) mod 7 表示明天。同样的道理,(w-1+7) mod 7 可以表示昨天星期数。

2.4.2 位运算符

位运算符对操作数进行二进制的位运算,位运算有 and(与)、or(或)、not(非)、xor(异或)、shl(左移)、shr(右移)。位运算的对象是整数。位运算的真值表如表 2-6 所示。

表 2-6 位运算的真值表

a	b	not a	a and b	a or b	a xor b
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

【例 2-5】 请看下面的位运算结果。

not 6 的结果为-7,因为 not 00000110 结果为 11111001(表示-7)。

8 and 7 的结果为 0,因为 00001000 and 00000111 的值为 00000000(表示 0)。

6 or 3 的结果是 7,因为 00000110 or 00000011 的值为 00000111(表示 7)。

6 xor 3 的结果是 5,因为 00000110 xor 00000011 的值是 00000101(表示 5)。

9 shl 1 的结果是 18,相当于 9 乘以 2,因为 00001001 移动后为 00010010(表示 18)。

27 shr 2 的结果是 6,相当于 27 div 2^2。

2.4.3　字符串运算符

字符串运算符用于将多个字符串或者字符连接起来。运算符为“+”，运算格式为：

<字符串表达式>|<字符表达式>+<字符串表达式>|<字符表达式>

当多个字符串连接起来的时候，后面一个字符串直接添加到前面一个字符串的后面，字符串的长度变长。例如：

```
'1+2'+'=3'                    //连接后变成'1+2=3'
'聪明的程序员使用'+'Delphi!'    //连接后变成'聪明的程序员使用 Delphi!'
```

2.4.4　关系运算符

关系运算符用于数值型、布尔型、字符型、字符串型数据的比较，运算结果是布尔型，关系运算符有：=、>、>=、<、<=、<>、in。字符比较依据是字符 ASCII 码的大小，字符串的比较是从左到右进行的。

例如：

```
2.0> 3.0          //值为 False
'A'<'a'           //值为 True
1>=2              //值为 False
3<>3.5            //值为 True
'ab'>'ac'         //值为 False,从左到右比较,前一个字符相同,故比较后面的'b'和'c'
1 in [2,3]        //值为 False,in 用于判断一个元素是否在一个集合中
```

2.4.5　逻辑运算符

逻辑运算符又称为布尔运算符，用于对布尔类型的数据进行运算。主要有 not(非)、and(与)、or(或)、xor(异或)。

逻辑运算的真值表如表 2-7 所示。

表 2-7　逻辑运算的真值表

a	b	not a	a and b	a or b	a xor b
False	False	True	False	False	False
False	True	True	False	True	True
True	False	False	False	True	True
True	True	False	True	True	False

布尔运算具有短路计算功能，即从左往右计算，如果表达式的值已经确定，则立即终止计算，不再继续计算其他表达式的值。例如：

```
var a:boolean;
false and a;      //值为 False,不会计算 a 的值
true or a;        //值为 True,不会计算 a 的值
```

2.4.6　运算符的优先级

运算符是有优先级的，优先级用于决定表达式中运算的先后次序，优先级高的先运

算，优先级低的后运算，同一优先级的按照从左到右的顺序运算。各种运算符按照从高到低的优先级顺序列出如下：

(1) 括号()

(2) 函数

(3) not、+、-(取正、负符号)

(4) 乘法类：*、/、div、mod、and、shl、shr

(5) 加法类：+、-、or、xor

(6) =、>、>=、<、<=、<>、in

建议在复杂的表达式中人为地加上括号以便提高表达式的可读性。

2.5 常用函数与过程

Delphi 提供了丰富的程序库，其中包含了大量的标准函数和过程。这些函数和过程都是在 SysUtils 单元中定义的，而 SysUtils 单元会自动被其他程序单元引用，因此用户无须在 uses 语句之后引用该单元。

掌握这些标准函数将大大降低编程的复杂度，本节将对比较常用的标准函数作介绍。

2.5.1 数学运算函数与过程

数学函数用于各种数学运算，主要有如下一些数学函数。

(1) 绝对值函数

函数格式为：

```
function abs(x);
```

说明：返回参数的绝对值。

(2) 平方函数

函数格式为：

```
function sqr(x: Extended): Extended;
function sqr(x: Integer): Integer;
```

说明：两种重载形式，一般使用第一种，参数为实型表达式。返回参数的平方即 x * x。

(3) 平方根函数

函数格式为：

```
function sqrt(x: Extended): Extended;
```

说明：返回非负数 x 的平方根。

(4) 三角函数

正弦函数的格式为：

```
function sin(x: Extended): Extended;
```

余弦函数的格式为：

```
function cos(x: Extended): Extended;
```

反正切函数的格式为：

```
function arctan(x: Extended): Extended;
```

说明：

① 正弦、余弦函数的参数是弧度，利用正弦、余弦可以计算出正切、余切。

② arctan 返回正切值为 x 的角度(单位是弧度)。

③ 利用 arctan 以及数学关系可以求出其他反正弦、反余弦函数。

下面的公式成立：

```
tan(x)=sin(x)/cos(x)
ctan(x)=cos(x)/sin(x)
arcsin(x)=arctan(x/sqrt(1-sqr(x)))
arccos(x)=arctan(sqrt(1-sqr(x))/x)
arcctan(x)=π/2-arctan(x)
```

(5) 取整函数

取整函数分为四舍五入取整和返回最大的不大于该数的整数的取整。四舍五入取整函数格式为：

```
function round(x: Extended): Int64;
```

说明：对 x 四舍五入取整。

返回最大的且不大于 x 本身的取整函数的格式为：

```
function trunc(x: Extended): Int64;
```

说明：返回最大的且不大于 x 本身的整数。

(6) 指数函数

指数函数的格式为：

```
function exp(x: Real): Real;
```

说明：返回 e^x。

(7) 对数函数

对数函数的格式为：

```
function ln(x: Real): Real;
```

说明：返回自然对数 ln(x)。

下列公式成立：

```
log_a b= log_e b/log_e a= ln(b)/ln(a)
lg(x)= ln(x)/ln(10)
x^n= exp(n * ln(x))
```

(8) 随机函数

随机函数的格式为：

```
function random [(Range: Integer)];
```

说明：

① 返回一个大于或者等于 0 且小于 Range(Range 是整数)的随机整数。

② 无参数用法，a＝random，a 的取值范围是 0≤a＜1。

(9) 随机过程

随机过程的格式为：

```
procedure randomize;
```

说明：随机种子，可以让 random 产生的随机数没有规律。

(10) π 函数

π 函数的格式为：

```
function pi: Extended;
```

说明：返回 π 值 3.1415926535897932385。

【例 2-6】 请看下面的函数举例。

(1) 随机产生一个大小在 97～122 之间的随机正整数 n。

解：

```
randomize;
n:=97+round(random(26));
```

或者：

```
randomize;
n:=97+round(random * 26);
```

(2) 求 sin(60)。

解：

```
n:=sin(60 * pi/180);
```

(3) 求 arcctan(1)。

解：

```
arcctan(1):=pi/2-arctan(1);          //值为 0.785398163397448,即 pi/4
```

(4) 求 $\log_3 5$。

解：

```
n:=ln(5)/ln(3);                      //值为 1.46497352071793
```

(5) 求 15^4。

解：

```
n:=exp(4 * ln(15));                    //值为 50625
```

2.5.2　字符处理函数与过程

(1) 大小写转换函数

函数格式为：

```
function UpperCase(const s: string): string;
```

说明：用于将 s 字符串中的所有小写字母转化为大写字母，其他字符不变。

```
function LowerCase(const s: string): string;
```

说明：用于将 s 字符串中的所有大写字母转化为小写字母，其他字符不变。

(2) 比较字符串大小的函数

函数格式为：

```
function CompareStr(const s1, s2: string): Integer;
```

说明：s1＞s2 返回大于 0 的数，s1＝s2 返回 0，s1＜s2 返回负数。

(3) 将一个或者多个字符串连接起来的函数

函数格式为：

```
function Concat(s1 [, s2,…, sn]: string): string;
```

说明：将 s1，s2，…，sn 依次连接起来。

(4) 查找字符串函数

函数格式为：

```
function Pos(substr: string; s: string): Integer;
```

说明：返回 substr 字符在 s 字符串中的位置，如果找不到 substr 则返回 0。

(5) 取自字符串函数

函数格式为：

```
function Copy(s; Index, count: Integer): string;
```

说明：在 s 中从第 index 位置开始取长度为 count 的字符串。如果 count 大于 s 字符串长度，则取空串；如果从 index 开始到最后不足 count 个字符，则取从 index 开始的所有字符。

(6) 求字符串长度的函数

函数格式为：

```
function Length(s): Integer;
```

说明：返回 s 字符串的长度。

(7) 去掉字符串前置不可见字符的函数

函数格式为：

```
function TrimLeft(const s: string): string; overload;
```

```
function TrimLeft(const s: WideString): WideString; overload;
```

说明：去掉 s 字符串的前置不可见字符，然后再返回 s。s 可以是字符型或者是宽字符型。小于或者等于 #32 的字符看作不可见字符。

(8) 去掉字符串右边不可见字符的函数

函数格式为：

```
function TrimRight(const s: string): string; overload;
function TrimRight(const s: WideString): WideString; overload;
```

说明：去掉 s 字符串右边不可见字符，然后再返回 s。小于或者等于 #32 的字符看作不可见字符。

(9) 去掉字符串前置和后面的不可见字符的函数

函数格式为：

```
function Trim(const s: string): string; overload;
function Trim(const s: WideString): WideString; overload;
```

说明：去掉 s 字符串前面和后面的不可见字符，再返回 s 字符串。

(10) 删除子字符串的过程

格式为：

```
procedure Delete(var s: string; index, count:Integer);
```

说明：用来在字符串 s 中删除从 index 开始、count 个字符长的子字符串。如果 index 大于字符串长度，则不删除字符；如果从 index 到字符结尾不足 count 个字符，则删除 index 之后的所有字符。

(11) 插入字符串的过程

格式为：

```
procedure Insert(source: string; var s: string; index: Integer);
```

说明：将 source 字符串插入到 s 中的第 index 字符处。

2.5.3 日期时间函数与过程

(1) 日期时间函数

日期时间函数的格式为：

```
function Now: TDateTime;
```

说明：返回当前的日期时间。例如，edit1. Text := datetimetostr(now)语句会显示类似“2006-8-18 10:38:54”的日期时间。

(2) 日期函数

日期函数的格式为：

```
function Date: TDateTime;
```

说明：返回当前的日期。例如，edit1. Text := datetimetostr(Date)或者 edit1. Text :=

datetostr(Date)语句会显示类似“2006-8-18”的日期。

(3) 时间函数

时间函数的格式为：

```
function Time: TDateTime;
function GetTime: TDateTime;
```

说明：返回当前的时间。例如，edit1. Text := timetostr(time)或者 edit1. Text := timetostr(gettime)语句会显示类似“10:46:50”的时间。

(4) 星期函数

星期函数的格式为：

```
function DayOfWeek(Date: TDateTime): Integer;
```

说明：返回星期几，星期天是一个星期的第 1 天，返回值是 1，星期 6 是一个星期的第 7 天，返回值 7，其他类推。例如，edit1. Text := inttostr(dayofweek(date))(假设当前日期是 2006 年 8 月 18 日)，返回 6，因为 2006 年 8 月 18 日是星期五。

(5) 日期与时间格式的函数

函数格式为：

```
function FormatDateTime(const format: string; DateTime: TDateTime): string;
```

说明：FormatDateTime 将指定的日期时间值按照 format 指定的格式输出。format 格式化字符串的取值和含义参见表 2-8。

表 2-8　format 格式化字符串的取值和含义

取　值	含　义
c	以短时间格式显示时间，即全部是数字的表示，FormatDateTime('c', now)输出为 2006-8-18 11:10:46
d	对应于时间中的日期，日期是一位则显示一位，是两位则显示两位
dd	对应于时间中的日期，始终显示两位日期，如 08
ddd	显示的是星期几，如 edit1. Text := FormatDateTime('ddd', now)语句显示星期五(假设日期是 2006 年 8 月 18 日)
dddd	与 ddd 相同，ddd 为短格式，dddd 为长格式
ddddd	长时间格式显示年、月、日，例如 edit1. Text := FormatDateTime('ddddd', now)语句显示“2006-8-18”
m	将月份显示为 1～12 格式
mm	将月份显示为 01～12 格式，有前置 0
mmm	显示类似“八月”，例如 edit1. Text := FormatDateTime('mmm', now)语句显示“八月”
mmmm	同上，mmm 为短格式，mmmm 为长格式
yy	将年份显示为 00～99
yyyy	将年份显示为 0000～9999

续表

取 值	含 义
h	将小时显示为 0～23
hh	将小时显示为 00～23
n	将分钟显示为 0～59
nn	将分钟显示为 00～59
s	将秒钟显示为 0～59
ss	将秒钟显示为 00～59
t	短格式显示时间
tt	长格式显示时间
am/pm	使用 12 小时制显示时间，am 表示上午，pm 表示下午
a/p	使用 12 小时制显示时间，a 表示上午，p 表示下午
ampm	上午显示“上午”，下午显示“下午”。例如，edit1. Text := FormatDateTime('ampm', now)显示“2006-8-18 11:38:01 上午”
/	日期分隔符号“-”。例如，edit1. Text := FormatDateTime('yy/mm/dd', now)显示结果为“06-08-18”
:	时间分隔符号“:”。例如，edit1. Text := FormatDateTime('hh:nn:ss', now)显示结果为“11:48:41”
'x'/ "x"	引号中的内容原样输出。例如，执行语句 edit2. Text := FormatDateTime('“现在是北京时间”hh:nn:ss', now) 输出结果是“现在是北京时间 11:57:50”

2.5.4 类型转换函数与过程

(1) 日期转化为字符串

日期转化为字符串的函数格式为：

```
function DateToStr(Date: TDateTime): string;
```

(2) 时间转化为字符串

时间转化为字符串的函数格式为：

```
function TimeToStr(Time: TDateTime): string;
```

(3) 时间、日期转化为字符串

将时间、日期转化为字符串的函数格式为：

```
function DateTimeToStr(DateTime: TDateTime): string;
```

(4) 时、分、秒、毫秒转化为时间类型

将时、分、秒、毫秒转化为时间类型的函数格式为：

```
function EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime;
```

(5) 时间类型转化为时、分、秒、毫秒

时间类型转化为时、分、秒、毫秒的过程格式为：

```
procedure DecodeTime(Time: TDateTime; var Hour, Min, Sec, MSec: Word);
```

说明：将时间日期型数据转化为时、分、秒、毫秒，Hour、Min、Sec、MSec 是 Word 类型的变量。

(6) 年、月、日转化为日期类型

将年、月、日转化为日期类型的函数格式为：

```
function EncodeDate(Year, Month, Day: Word): TDateTime;
```

(7) 日期类型转化为年、月、日

将日期类型转化为年、月、日的过程格式为：

```
procedure DecodeDate(Date: TDateTime; var Year, Month, Day: Word);
```

说明：Year、Month、Day 是 Word 类型的变量。

(8) 字符串型转化为数值型

字符串型转化为数值型的过程格式为：

```
procedure Val(s; var v; var code: Integer);
```

说明：将字符串 s 转化为数值型的 v，code 为 0 则转化成功，否则转化出错。

(9) 整型转化为字符串型

整型转化为字符串型的函数格式为：

```
function IntToStr(Value: Integer): string; overload;
function IntToStr(Value: Int64): string; overload;
```

说明：整数可以是 Integer 型或 Int64 型。

(10) 字符串型转化为整型

字符串型转化为整型的格式为：

```
function StrToInt(const s: string): Integer;
function StrToInt64(const s: string): Int64;
```

说明：可以将字符串转化为 Integer 型或 Int64 型。

(11) 浮点型转化为字符串型

浮点型转化为字符串型的函数格式为：

```
function FloatToStr(Value: Extended): string;
```

(12) 字符串型转化为浮点型

字符串型转化为浮点型的函数格式为：

```
function StrToFloat(const s: string): Extended;
```

(13) 字符串型转化为布尔型

字符串型转化为布尔型的函数格式为：

```
function StrToBool(const S: string): Boolean;
```

(14) 布尔型转化为字符串型

布尔型转化为字符串型的函数格式为：

```
function BoolToStr(b: Boolean; UseBoolStrs: Boolean=False): string;
```

说明：如果 b 是 True 则函数返回 True 或者－1，UseBoolStrs 为 True 时返回字符串 True，UseBoolStrs 为 False 时返回－1。如果 b 是 False 则返回 False 或者 0，UseBoolStrs 为 True 时返回字符串 False，UseBoolStrs 为 False 时返回 0。例如：Edit1. Text := BoolToStr(CheckBox1. Checked, CheckBox2. Checked)。

2.5.5 其他函数与过程

(1) ASCII 码转化为字符

格式为：

```
function Chr(x: Byte): Char;
```

说明：将 x 转化为字符，如 chr(97)为 a 字符。

(2) 取序号函数

格式为：

```
function Ord(x);
```

说明：适用于顺序类型，如整型、字符型、布尔型、枚举型和子界型。例如，1 的序号为 1，a 的序号是 a 的 ASCII 码值 97，False 的序号是 0，True 的序号是 1 等。

(3) 前导函数

格式为：

```
function Pred(x);
```

说明：求前导，适用于整型、字符型、布尔型、枚举型和子界等顺序类型。例如，b 的前导是 a，2 的前导是 1，True 的前导是 False 等。

(4) 后继函数

格式为：

```
function Succ(x);
```

说明：求后继，适用于整型、字符型、布尔型、枚举型和子界等顺序类型。例如，b 的后继是 c，2 的后继是 3，False 的后继是 True 等。

2.6 语句

语句是执行计算机程序的指令，计算机的各种操作都是通过计算机语句来实现的。Object Pascal 语句主要分为基本语句和构造语句两种。语句以“;”作为结束标志，end 之

前的“;”可以省略不写。基本语句分为赋值语句、读写语句、转向语句和空语句，构造语句分为复合语句、条件语句、循环语句和开域语句。

基本语句是执行一项特定操作的简单语句，本节讲述基本语句。构造语句将在其他章节讲解。

1. 赋值语句

将一个表达式的值赋给一个变量、数组元素或者对象的属性，赋值号为“:=”。

例如：

```
var i;                                  //定义变量 i
...
i:=12;                                  //变量 i 取值 12
```

再如：

```
var st:string;
...
st:='聪明的程序员使用 Delphi!';        //给 st 字符串变量赋值
label1.Caption:=st;                     //给属性赋值
```

2. 空语句

单独一个分号就是一个空语句。设计空语句的目的只是语法需要而已，空语句不会做任何操作。

例如：

```
var i;                          //定义变量 i
...
i:=12; ;                        //变量 i 取值 12,第 1、2 个分号之间就是一个空语句
```

3. 读语句

读语句用于从标准设备(键盘)输入数据。它有两种格式：

```
read(变量,变量,…);
readln(变量,变量,…);
```

其中，read 语句必须有参数，而 readln 可以没有参数，没有参数的 readln 语句不加括号，单独一个 readln 语句表示等待输入回车。

例如：

①

```
var i,j,k;
read(i);
readln(j,k);                    //最后两个 read 语句等价于 readln(i,j,k);
```

②

```
var ch1,ch2:char;
readln(ch1,ch2);                //输入的两个字符中间不要加任何分隔符,可以写成两个读语句
```

③

```
var ch1:char; i:integer;
readln(i,ch1);                    //错,这样写无法输入 ch1
readln(ch1,i);                    //对,在输入字符后
```

4. 写语句

写语句用于将数据从标准设备(显示器)输出,有两种格式:

```
write(项 1,项 2,…);
writeln(项 1,项 2,…);
```

其中,项可以是常数、变量、函数、表达式等。write 语句必须有参数,write 语句和 writeln 语句用于将这些项输出在显示器上,writeln 表示输出后换一行。writeln 语句可以没有参数,表示输出一个空行,没有参数的 writeln 语句不需要括号。

根据美观需要,每项的后面还可以定义场宽,格式为:

```
:整数表达式
```

说明:如果项所占的数据位数不够,前面可以加空格;如果项所占的数据位数太宽,可以突破场宽的限制,按照项的实际宽度显示。

如果项是实数,还可以这样规定该项所占的场宽:

```
:整数表达式 1:整数表达式 2
```

整数表达式 1 表示实数总宽度,整数表达式 2 表示小数部分宽度。

【例 2-7】 求一个三位数的各位数字。

分析:本例将用到 write、read 等语句,用于输入、输出数据。

步骤如下:

(1) 选择菜单 File|New|Others...,出现新建项对话框,在对话框中选择 Console Application,按 OK 按钮,出现代码编辑器。

(2) 在代码编辑器中输入代码:

```
program Project1;

{$APPTYPE CONSOLE}

uses
  SysUtils;
 var a,gw,sw,bw:integer;

begin
 write('请输入一个三位数=');                    //提示输入一个三位数
 readln(a);                                      //输入一个数
 gw:=a mod 10;                                   //求个位数字
 sw:=(a div 10)mod 10;                           //求十位数字
 bw:=a div 100;                                  //求百位数字
 writeln('个位数字是',gw,',十位数字是',sw,',百位数字是',bw);
```

```
 readln                                          //在等待输入回车的过程中看结果
end.
```

(3) 按 F9 键运行,结果如下:

```
请输入一个三位数=236                              // 波浪线表示键盘输入,输入 236 后回车
个位数字是 6,十位数字是 3,百位数字是 2
```

5. goto 语句

goto 语句是转向语句,与 goto 同类的转向语句还有 break、continue,这两个转向语句将在流程控制一章讲解。goto 语句不符合结构化程序编程的思想,建议少用,但是有时候使用 goto 语句可以使程序变得非常简单、方便。

goto 语句的格式为:

```
label 语句标号 1,语句标号 2,…;                    //在定义部分定义语句标号
goto 语句标号 n;                                  // 转向到语句标号 n 指向的行执行
⋮
语句标号 n:语句 x;                                //转到此处执行
```

说明:

(1) 语句标号要先定义后使用。

(2) 不允许用 goto 语句从构造语句(循环、分支等)外转到构造语句内。

6. 复合语句

根据语法需要,例如,如果某个条件成立则需要执行多个语句,这时候可将多个语句用 begin 和 end 括起来,形成复合语句。复合语句的格式为:

```
begin
  <语句 1>;
  ⋮
  <语句 n>;
end;
```

2.7　小结

本章讲述了 Object Pascal 语言,包括以下知识点:语言成分、数据类型、常量和变量、运算符和表达式、常用的函数和过程以及语句。

运算符和表达式是本章的重点,常用的函数和过程也是本章的重点,读者务必学会这些函数与过程,并能够在自己的程序中灵活地使用它们。

习题

1. 下列哪些符号可以作为用户自定义的标识符?

(1) begin　(2) teacher　(3) 4an　(4) program　(5) x+y

(6) xy (7) ord (8) sqr

2. 在 Object Pascal 中哪些是标准数据类型？哪些是顺序类型？

3. 下列哪些表达式是常量？哪些表达式是变量？

(1) x+y (2) name (3) false (4) 'name' (5) '3+4=5'

(6) 4.5 (7) else (8) 5E4.5 (9) 12.34 (10) sin90

4. 下列哪些常量的定义是合法的？

(1) Const c :=(3>4) (2) Const c=3.4 (3) Const c :=10 and 20

(4) Const f=false (5) Const g:real=4.6 (6) Const k=3+4

5. 下列变量的定义哪些是合法的？

(1) var a,b,c:integer;
a:real; (2) var f1,f2:boolean;
var f3,f4:integer; (3) var a:string[8]

6. 计算下列表达式的值。

(1) 20+19 div −3 (2) succ('b')+pred('c') (3) chr(13)

(4) −30 mod 4 (5) ('a'>'b') and false (6) exp(ln(3))

7. 将下列数学表达式改写成等价的 Object Pascal 表达式。

(1) $\dfrac{1+\dfrac{y}{x}}{1-\dfrac{y}{x}}$ (2) $x^2+\sqrt[3]{1+\dfrac{x^2}{y}}$ (3) $e^x \lg(a+b)(-2t^2)$

(4) ctan(30)×arcctan(1)

8. 如何随机产生一个大写英文字母？

9. 如何随机产生一个三位的正整数？

10. 如何对 x 小数点后的第 3 位进行四舍五入？

11. 下列赋值语句正确的有哪些？

(1) sin90 :=23; (2) x+y :=78; (3) sqrt :=87;

(4) a :=true<false; (5) pi :=90;

第3章

窗体和基本组件

在程序设计中，程序界面是非常重要的，界面是否美观、友好是给用户的第一印象。程序界面元素主要有窗体和基本组件。窗体是应用程序的操作界面，程序的操作元素都必须添加到窗体上，没有窗体程序的框架就无法建起。而基本组件则是用户与应用程序交互的工具，有了这些基本的常见组件用户才能够方便地操作程序，程序才能够响应用户的请求。

3.1 窗体

窗体是应用程序的操作界面，用于存放组件，没有窗体应用程序就无法搭建。窗体由标题栏、工作区、边界构成。标题栏不仅有控制菜单，还有最小化、最大化(恢复)、关闭等按钮，窗体的标题栏可以用于改变窗体的大小、位置，还可以进行最大化、最小化、恢复窗体的大小位置等操作。可以在窗体标题栏中显示窗体的标题文字。边框可以用于调整窗体的大小。工作区是窗体存放组件的位置。

1. 窗体的属性

窗体的属性用来描述窗体的高低、长宽、颜色、边框，主要属性如下。

(1) Name 属性和 Caption 属性

Name 属性：对象的名称属性，它用来唯一标识对象，一个程序中不同对象的名称是不能够相同的。系统是根据对象的名称来识别不同对象的。对象的名称最好要做到见名知义，如 Form1 表示窗体，而 Edit1 表示文本框。

Caption 属性：窗体的标题文字，是字符类型。默认情况下，窗体的 Caption 属性与 Name 属性是相同的。可以修改窗体的标题文字，例如，设置窗体的标题文字为“游戏程序”。

(2) Height、Width、ClientHeight、ClientWidth、Top 和 Left 属性

Height、Width 分别表示窗体的高度和宽度。ClientHeight、ClientWidth 分别表示窗体工作区的高度和宽度(工作区是不包括标题栏和边框的)。Top 和 Left 分别表示窗体左上角在屏幕中的垂直和水平位置。

(3) Enabled 属性

窗体是否有效，值为 True 时窗体有效，此时窗体可以响应各种事件；值为 False 时窗

体无效，窗体不响应事件。

(4) Visible 属性

窗体是否显示，值为 True 时窗体显示，值为 False 时窗体不显示。

(5) Color 属性

窗体的颜色属性，可以使用 rgb 函数来表示，也可以使用类似的 clbackground 等枚举值。

(6) Font 属性

窗体的字体属性，包括字的大小、颜色、下划线、删除线、粗体和斜体等。单击属性 Font 后面的省略号按钮，出现系统设置字体对话框。

(7) Align 属性

Align 属性用来决定窗体在屏幕中的对齐方式，该属性是枚举类型，其具体取值及其含义见表 3-1。

表 3-1 Align 属性的取值及其含义

取 值	说 明
alBottom	窗体位于屏幕下方，宽度和屏幕一样(Width 属性无效)
alClient	窗体在屏幕中间，Height 和 Width 属性无效
alLeft	窗体在屏幕左边，高度和屏幕一样(Height 属性无效)
alRight	窗体在屏幕右边，高度和屏幕一样(Height 属性无效)
alNone	窗体在原始位置(Height 和 Width 属性都有效)
alTop	窗体在屏幕上方，宽度和屏幕一样(Width 属性无效)

(8) FormStyle 属性

FormStyle 属性用于确定窗体的类型，该属性是枚举类型，属性的具体取值及其含义见表 3-2。

表 3-2 FormStyle 属性的取值及其含义

取 值	说 明
fsNormal	普通的窗体
fsMDIChild	MDI 窗体(多文档窗体)的子窗体
fsMDIForm	MDI 主窗体
fsStayOnTop	在桌面顶层的窗体

(9) BorderIcon 属性

BorderIcon 属性用于设置窗体的控制按钮和菜单，是集合类型，各个元素及其含义见表 3-3。

表 3-3　BorderIcon 属性的元素及其含义

元　素	说　明
biSystemMenu	窗体标题栏含有系统菜单(左边)
biMinimize	窗体标题栏含有最小化按钮(右边)
biMaximize	窗体标题栏含有最大化按钮(右边)
biHelp	窗体标题栏含有帮助按钮(此时无最小化、最大化按钮)

(10) BorderStyle 属性

BorderStyle 属性用于确定窗体边框的类型,BorderStyle 属性的取值及其含义见表 3-4。

表 3-4　BorderStyle 属性的取值及其含义

取　值	说　明
bsNone	窗体无边框线,无法改变窗体大小
bsSingle	窗体边框是单线,无法改变窗体大小
bsDialog	对话框边框,无法改变窗体大小
bsSizeable	有边框,可以改变窗体大小
bsToolWindow	和 bsSingle 类似,只是标题栏稍小,无法改变窗体大小
bsSizeToolWin	和 bsSizeable 类似,只是标题栏稍小,可以改变窗体大小

(11) WindowStyle 属性

WindowStyle 属性用于设置窗体运行时的显示状态,是枚举类型,取值及其含义见表 3-5。

表 3-5　WindowStyle 属性的取值及其含义

取　值	说　明
wsNormal	窗体的位置和大小由设计时决定
wsMinimized	运行时,窗体最小化
wsMaximized	运行时,窗体最大化

2. 窗体的事件

窗体的事件很多,常用的事件如下。

(1) OnCreate 事件

建立窗体时首先触发该事件,一般把程序的初始化代码写在 OnCreate 事件过程中。

(2) OnShow 事件

显示窗体时,触发该事件。

(3) OnPaint 事件

重画窗体事件,窗体改变大小,在窗体上移动其他窗体时触发该事件。

(4) OnActivate 事件

激发窗体的时候(窗体得到焦点)触发该事件。

(5) OnClose 事件和 OnCloseQuery 事件

关闭窗体时触发这两个事件。当窗体要关闭时,首先触发 OnCloseQuery 事件,该事件处理过程中有一个 Boolean 类型的参数 CanClose,默认值为 True,表示窗体可以关闭,当 CanClose 的值为 False 时窗体不能关闭,不会触发 OnClose 事件。如果在 OnCloseQuery 事件过程中设置 CanClose 的值为 False 则窗体可以关闭,将触发 OnClose 事件。在 OnClose 事件处理过程中,有一个参数 Action 用来决定关闭窗体的实际操作。Action 的取值和含义见表 3-6。

表 3-6 Action 的取值和含义

取 值	说 明
caNone	窗体不关闭,无任何操作
caHide	窗体不关闭,但是隐藏,仍然在运行
caFree	关闭窗体,窗体不再在内存运行
caMinimize	不关闭窗体,最小化窗体

3. 窗体的方法

窗体的方法很多,主要的常用方法如下。

(1) Release 方法

从内存释放窗体。

(2) Show 方法

显示窗体。

(3) Hide 方法

隐藏窗体。

(4) Close 方法

用于关闭窗体。例如,Form1.Close 表示关闭窗体,并触发 OnClose 事件和 OnCloseQuery 事件。

4. 窗体的应用举例

【例 3-1】 要求程序运行时窗体位于屏幕中央,请编写程序。

本程序应该在窗体的 OnFormCreate 事件过程中写代码,代码如下:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 Form1.Left:=(1024-form1.Width)div 2;
 form1.Top:=(768-form1.Height)div 2        //假设屏幕分辨率为 1024×768
end;
```

【例 3-2】 有时候用户关闭程序时弹出对话框提示是否关闭窗口,此时需要用到 Form 的 OnClose 事件和 OnCloseQuery 事件。请编写程序完成。

本程序可以使用 OnCloseQuery 事件来完成。先添加按钮 Button1，编写 Button1 的 OnClose 事件过程如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 form1.Close;          //该方法将首先触发 OnCloseQuery 事件
end;
```

再编写 Form1 的 OnCloseQuery 事件过程，代码如下：

```
procedure TForm1.FormCloseQuery(Sender: TObject; var CanClose: Boolean);

 var i:word;

begin
 i:=messagedlg('是否关闭窗口?',mtconfirmation,[mbyes,mbno],1);
 if i=mryes                    //单击"是"按钮
 then canclose:=true           //关闭窗口,将触发 OnClose 事件
 else canclose:=false          //不关闭窗口,不会触发 OnClose 事件
end;
```

说明：单击 Button1 按钮或者直接单击窗体右上角的关闭按钮将触发 Form1 的 OnCloseQuery 事件，程序提示"是否关闭窗口?"，选择 yes 可以关闭窗口，选择 no 不关闭窗口。

上面的程序还可以不使用 Form1 的 OnCloseQuery 事件，而使用 Form1 的 OnClose 事件。可将上面 Form1 的 OnCloseQuery 事件过程改成 Form1 的 OnClose 事件过程，代码如下：

```
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);

var i:word;

begin
 i:=messagedlg('是否关闭窗口?',mtconfirmation,[mbyes,mbno],1);
 if i=mryes
 then Action:=cafree           //关闭窗口
 else Action:=canone           //不关闭窗口
end;
```

说明：单击 Button1 按钮或者直接单击窗体右上角的关闭按钮，此时触发 Form1 的 OnCloseQuery 事件。由于 Form1 的 OnCloseQuery 事件过程中参数 CanClose 的默认值是 True，因此程序触发 Form1 的 OnClose 事件。程序提示"是否关闭窗口?"，选择 yes 可以关闭窗口，选择 no 不关闭窗口。

3.2　文本显示与编辑组件

窗体是应用程序的操作界面，用于存放组件，只有在窗体中加入其他组件，用户才可以和程序交互。标签、单行文本框、多行编辑框、格式编辑框就是这类用于人机交互的组件。

3.2.1 Label 组件

Label 标签是最常见的文本显示组件，与文本框不同的是，文本框可以编辑，标签运行之后就不能够直接编辑了，因此它主要用于静态的文本显示。Label 组件位于 Standard 组件面板中。

1. Label 的属性

(1) Caption 属性：标签上显示的文字，如果 Caption 属性中含有字符“&”，那么该字符之后的第一个字符为加速键。

(2) AutoSize 属性：Boolean 型，决定标签是否随文字的变化而改变大小。

(3) Alignment 属性：标签的对齐方式，有 3 个枚举值，分别是左对齐、居中对齐、右对齐。

(4) Layout 属性：标签的对齐方式，有 3 个枚举值，分别是上对齐、居中对齐、下对齐。

(5) WordWrap 属性：Boolean 型，是否折行显示。

(6) Transparent 属性：Boolean 型，背景是否透明。

(7) FocusControl 属性：按下加速键时，获得焦点的组件。

2. Label 的应用

【例 3-3】 编写一个密码登录框程序。

程序设计步骤：

(1) 程序可以使用标签、编辑框、命令按钮等，界面设计如图 3-1 所示。

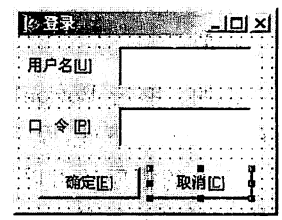


图 3-1 登录程序界面

(2) 各个组件的属性设置见表 3-7。

表 3-7 登录程序的属性设置

对 象	属 性	属 性 值	说 明
Label1	Name	Label1	标签名称
	Caption	用户名[&U]	设置标题和加速键
	FocusControl	Edit1	按下 Alt+U 键时 Edit1 获得焦点
Label2	Name	Label2	标签名称
	Caption	口令[&P]	设置标题和加速键
	FocusControl	Edit2	按下 Alt+P 键时 Edit2 获得焦点
Edit1	Name	Edit1	编辑框名称
	Text	空	编辑框为空
Edit2	Name	Edit2	编辑框名称
	Text	空	编辑框为空
	Passwordchar	*	输入的密码显示为“*”

续表

对　象	属　　性	属　性　值	说　　明
Button1	Name	Button1	按钮名称
	Caption	确定[&E]	设置标题和加速键
Button2	Name	Button2	按钮名称
	Caption	取消[&C]	设置标题和加速键

(3) 编写程序如下：

“确定”按钮 Button1 的 OnClick 事件过程：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 if (uppercase(edit1.Text)='ABCDEF') and (edit2.Text='123')
 then showmessage('欢迎使用本系统')
 else begin
        showmessage('口令或者用户名错');
        edit1.Text:='';
        edit2.Text:='';
        edit1.SetFocus;
       end;
end;
```

“取消”按钮 Button2 的 OnClick 事件过程：

```
procedure TForm1.Button2Click(Sender: TObject);
begin
 edit1.Text:='';
 edit2.Text:='';
 edit1.SetFocus
end;
```

说明：函数 uppercase()可以将输入的小写英文字母变成大写英文字母。

3.2.2　Edit 组件

Edit 编辑框是最常见的文本输入、显示组件，它不仅可以输入、显示，而且还可以编辑、修改。Edit 组件位于 Standard 组件面板中。

1. Edit 组件的属性

(1) Text 属性：String 类型，文本框中的文本内容。

(2) AutoSelect 属性：Boolean 类型，设置当编辑框得到焦点的时候，是否自动选定文本。值为 True 时，当编辑框得到焦点时自动选定文本框中的文本；值为 False 时，当编辑框得到焦点时不选定文本框中的文本。

(3) Enabled 属性：Boolean 类型，编辑框是否有效。值为 True 时编辑框有效，值为 False 时编辑框无效。

(4) ReadOnly 属性：Boolean 类型，决定编辑框中的内容是否可以编辑。

(5) SelStart 属性：Integer 类型，选定文本首字符的位置，或者光标所在位置。编辑框的第一个字符的序号是 0。

(6) SelLength 属性：Integer 类型，被选定文本的长度。

(7) SelText 属性：String 类型，被选定的文本内容。如果设置了 SelStart 属性和 SelLength 属性，则自动会选定一段文字，文字的内容为 SelText。

(8) CharCase 属性：枚举类型，ecNormal 表示不改变输入字符的大小写，ecLowerCase 表示将输入字符中的英文转化为小写字符，ecUpperCase 表示将输入字符中的英文转化为大写字符。

(9) HideSelection 属性：Boolean 类型，选定的文字是否以加强的背景颜色显示。

(10) PasswordChar 属性：默认值是#0，此时，原样显示编辑框中的字符，如果将其设置为其他字符，如“*”，则编辑框中的所有字符都显示为“*”(虽然显示为“*”，但是 Text 属性仍然不会改变)，一般用于设置口令。

2. Edit 组件的方法

(1) SetFocus：置焦点。

(2) SelectAll：选定所有文本。

3. Edit 组件的主要事件

(1) OnChange 事件：编辑框的内容发生改变。

(2) OnEnter 事件：编辑框得到焦点。

(3) OnExit 事件：编辑框失去焦点。

(4) OnKeyPress 事件：按键事件，参数 Key 表示所按下的字符，它返回的是一个字符。

(5) OnKeyDown 事件：键盘按下事件，Key 为返回的按键(如 A 和 a 的 Key 都是 65)，还可以通过参数 Shift 检测 Shift、Ctrl、Alt 等键是否被按下。

(6) OnKeyUp 事件：键盘松开事件，参数与 OnKeyDown 类似。

可以将例 3-3 的程序改进一下，编写 Edit2 的 OnKeyPress 事件过程：

```
procedure TForm1.Edit2KeyPress(Sender: TObject; var Key: Char);
begin
 if key=#13 then          //如果按回车键
 begin
  if (uppercase(edit1.Text)='ABC') and (edit2.Text='123')
  then showmessage('欢迎使用本系统')
  else begin
         showmessage('口令或者用户名错');
         edit1.Text:='';
         edit2.Text:='';
         edit1.SetFocus;
        end;
 end;
end;
```

说明：改进后的程序在输入用户名和口令后无须按“确定”按钮，直接在 Edit2 中按

回车即可判断用户名和口令是否正确。

4. Edit 的应用

【例 3-4】 设计一个小学生加法练习器。练习器能随机给出两个两位数的正整数，要求写出两数之和，如果结果正确则提示“你真聪明”，如果答案错误则提示“错误，请重做”。请编写程序。

程序设计步骤：

（1）添加组件，在窗体中添加 Edit1、Edit2、Edit3、Label1、Label2、Button1 和 Button2。设置 Label1 的 Caption 为“＋”，Label2 的 Caption 为“＝”。设置 Button1 的 Caption 为“出题”，Button2 的 Caption 为“判断”。设置 3 个 Edit 的 Text 为空，并调整组件的位置和大小。界面如图 3-2 所示。

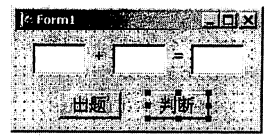


图 3-2　加法练习器界面

（2）编写程序如下：

```
procedure TForm1.Button1Click(Sender: TObject);

var x,y:integer;

begin
 randomize;
 x:=10+random(90);
 y:=10+random(90);                    //x,y是两位数
 edit1.Text:=inttostr(x);
 edit2.Text:=inttostr(y);
 edit3.Text:='';
 edit3.SetFocus;                      //Edit3置空,并置焦点等待输入结果
end;

procedure TForm1.Button2Click(Sender: TObject);

var z,i:word;

begin
 z:=strtoint(edit3.Text);
 if z=strtoint(edit1.Text)+strtoint(edit2.Text)
 then i:=messagedlg('你真聪明',mtinformation,[mbok],1)
 else begin
        i:=messagedlg('错误,请重做',mterror,[mbok],1);
        edit3.Text:='';
        edit3.SetFocus;               //错误后需要重做
      end;
 end;
```

说明：如果本程序要求每次运行都出题 10 次，出题 10 次后程序自动结束。这时可以使用属性 Tag，绝大多数组件都有 Tag 属性，其初始值为 0。一般可以把 Tag 看成一个整型的全局变量。修改后的程序如下：

```
procedure TForm1.Button1Click(Sender: TObject);

var x,y:integer;

begin
 randomize;
 x:=10+random(90);
 y:=10+random(90);              //x,y是两位数
 edit1.Text:=inttostr(x);
 edit2.Text:=inttostr(y);
 edit3.Text:='';
 edit3.SetFocus;                //Edit3置空,并置焦点等待输入结果
 form1.Tag:=form1.Tag+1;        //初始值为0,每出一道题Tag加1
end;

procedure TForm1.Button2Click(Sender: TObject);

var z,i:word;

begin
 z:=strtoint(edit3.Text);
 if z=strtoint(edit1.Text)+strtoint(edit2.Text)
 then i:=messagedlg('你真聪明',mtinformation,[mbok],1)
 else begin
        i:=messagedlg('错误,请重做',mterror,[mbok],1);
        edit3.Text:='';
        edit3.SetFocus;         //错误后需要重做
       end;
 if form1.Tag=10 then application.Terminate;
end;
```

说明：虽然读者还没有接触到变量作用域的相关知识，但是此处巧妙地利用 Tag 属性解决了类似变量作用域的问题。在程序的很多地方，读者都可以举一反三。

【例 3-5】 在编辑框中输入学号、姓名、性别、家庭住址等信息，按“提交”按钮后把这些信息显示在标签中。

程序设计步骤：

(1) 添加组件到窗体，包括 Label 标签组件 5 个、Edit 编辑框组件 4 个、Button 命令按钮组件 2 个。界面如图 3-3 所示。

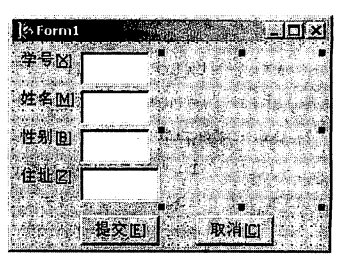


图 3-3 程序界面

(2) 设置这些组件的属性，参见表 3-8。

(3) 编写程序，同时选定 4 个编辑框，然后在 Object Inspector 的 Events 选项卡中，选定 OnEnter 事件，双击其右边的空格，右边空格显示 Edit1Enter(表示 4 个编辑框得到焦点都会执行 TForm1.Edit1Enter 过程)，并进入代码编辑器。编写如下代码：

表 3-8　属性设置

对　象	属　性	属 性 值	说　明
Label1	Caption	学号[&X]	设置标签标题和加速键
	FocusControl	Edit1	按下 Alt+X 键时 Edit1 获得焦点
Label2	Caption	姓名[&M]	设置标签标题和加速键
	FocusControl	Edit2	按下 Alt+M 键时 Edit2 获得焦点
Label3	Caption	性别[&B]	设置标签标题和加速键
	FocusControl	Edit3	按 Alt+B 键时 Edit3 获得焦点
Label4	Caption	住址[&Z]	设置标签标题和加速键
	FocusControl	Edit4	按 Alt+Z 键时 Edit4 获得焦点
Label5	Caption	空	标题为空
Edit1～Edit4	Font. Size	10	字号为 10
Edit1～Edit4	Text	空	编辑框为空
Button1	Caption	提交[&E]	设置标题和加速键
Button2	Caption	取消[&S]	设置标题和加速键

```
procedure TForm1.Edit1Enter(Sender: TObject);
begin
 (sender as tedit).color:=rgb(255,0,0);        //得到焦点的编辑框是红色
end;
```

同时选定 4 个编辑框，然后在 Object Inspector 的 Events 选项卡中，选定 OnExit 事件，双击其右边的空格，右边空格显示 Edit1Exit(表示 4 个编辑框得到焦点都会执行 TForm1. Edit1Exit 过程)，并进入代码编辑器。编写如下代码：

```
procedure TForm1.Edit1Exit(Sender: TObject);
begin
 (sender as tedit).color:=rgb(255,255,255);        //失去焦点的编辑框是白色
end;
```

“提交”按钮的代码：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 label5.Caption:='学号:'+edit1.Text ;
 label5.Caption:=label5.Caption+chr(13)+'姓名:'+edit2.Text;
 label5.Caption:=label5.Caption+chr(13)+'性别:'+edit3.Text;
 label5.Caption:=label5.Caption+chr(13)+'住址:'+edit4.Text;
end;
```

“取消”按钮的代码：

```
procedure TForm1.Button2Click(Sender: TObject);
```

```
begin
 edit1.Text:='';
 edit2.Text:='';
 edit3.Text:='';
 edit4.Text:='';
 edit1.SetFocus;
 label5.Caption:='';
end;
```

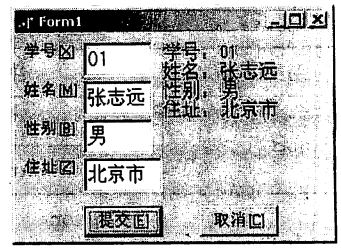


图 3-4 程序运行后的界面

(4) 按 F9 键运行，运行界面如图 3-4 所示。

3.2.3 Memo 组件

Edit 组件仅可以显示单行文本，如果要显示多行文本则显得无能为力，要处理多行文本需要使用 Memo 组件。

1. Memo 的属性

(1) Lines 属性：用于对 Memo 组件中的内容进行处理。单击 Lines 属性后的“省略号”按钮，打开 String List Editor 对话框，在其中输入多行字符串。

Lines 实际上是一个对象，它是一个数组，下标从 0 开始。例如：s := Memo1. Lines [3]；表示将 Memo1 中第 4 行文本内容赋值给变量 s。Lines 还有如下一些用法：

```
Memo1.Lines.Add('A New Line');                  //在最后添加一行
Memo1.Lines.Delete(4);                          //删除第 5 行
Memo1.Lines.Insert(1, 'Insert A New Line');     //在第 2 行插入一行
Memo1.Lines.Move(3,5);                          //将第 4 行移动到第 6 行
```

(2) WantReturns 属性：Boolean 属性。值为 True 时按 Enter 键插入一个回车符；值为 False 时按 Enter 键表示事件提交窗体处理，如果想输入回车符则需要按 Ctrl+Enter 键。

(3) WantTabs 属性：值为 True 时，按 Tab 键插入一个 Tab 字符，值为 False 时按 Tab 键将改变获得焦点的组件。

(4) ScrollBars 属性：控制 Memo 组件的滚动条，取值分别有 ssNone(无滚动条)、ssHorizontal(只有水平滚动条)、ssVertical(只有垂直滚动条)和 ssBoth(同时有水平与垂直滚动条)。

(5) Modified 属性：确定 Memo 组件是否被修改了。

2. Memo 的应用

【例 3-6】 在 Memo 中输入文字，按“提交”按钮后，文字显示在标签中。要求按回车键也可以将文本显示在标签中。请编写程序。

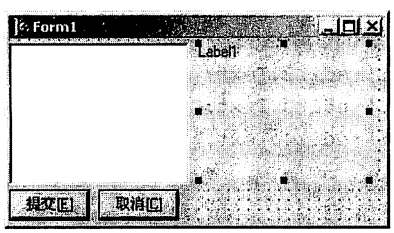


图 3-5 例 3-6 的用户界面

分析：应该将 Memo1 的 WantReturns 属性设置为 False，并且将“提交”按钮的 Default 属性设置为 True。其他组件的属性设置比较简单，程序代码也比较简单。

程序设计步骤：

(1) 添加组件 Button1、Button2、Memo1 和 Label1 到窗体，并设置组件的属性，界面如图 3-5 所示。

(2) 各个组件的属性设置见表 3-9。

表 3-9　属性设置表

对　象	属　性	属 性 值	说　明
Memo1	Name	Memo1	名称属性
	Lines	空	程序运行开始 Memo 为空
	WantReturns	False	按回车键,事件交由窗体处理
Button1	Name	Button1	“提交”按钮的名称属性
	Default	True	“默认”按钮
	Caption	提交[&E]	设置加速键
Button2	Name	Button2	“取消”按钮的名称属性
	Caption	取消[&C]	设置加速键
	Cancel	True	按 Esc 键等价于单击“取消”按钮
Label1	Name	Label1	标签的名称属性
	WordWrap	True	换行显示
	Caption	空	标签标题为空

(3) 编写程序如下:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 label1.Caption:=memo1.Text;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
 memo1.Text:='';
 label1.Caption:='';
end;
```

(4) 按 F9 键运行程序,在 Memo1 中输入文字然后按回车键相当于单击“提交”按钮,按“Ctrl+回车键”可以在 Memo 中插入换行符,按 Esc 键相当于单击“取消”按钮。程序正常运行界面如图 3-6 所示。

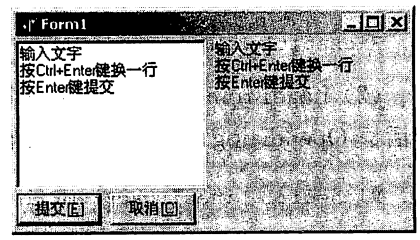


图 3-6　程序运行界面

说明:这里将 Memo1 的 WantReturns 设置为 False,并且设置 Button1 的 Default 为 True。这样设置后,在 Memo1 中输入内容后无须按 Button1(“提交”按钮),只需直接按回车键即可(相当于单击了 Button1 按钮)。

3.2.4 RichEdit 组件

RichEdit 组件是基于 Windows RTF(Rich Text Format)格式的文本，RichEdit 是包含各种编辑格式的多行文本编辑器，它可以设置不同文本的格式，如颜色、字体、字号、下划线、粗体、斜体、对齐方式等。RichEdit 组件位于 Win32 组件面板中。

1. 主要属性

(1) SelStart 属性、SelLength 属性、SelText 属性、WantReturns 属性、WantTabs 属性、ScrollBars 属性、Modified 属性、HideSelection 等属性。这些属性与 Edit 组件和 Memo 组件类似。

(2) SelAttributes 属性：选定文字的字体。下面的语句是将字体对话框选定的字体赋给 RichEdit1 中选定的文本。还有一些用法将在后面章节详细讲解。

```
Richedit1.SelAttributes.Assign(Fontdialog1.Font);
```

(3) DefAttributes 属性：RichEdit 默认字体。下面的语句是将 RichEdit1 默认的字体赋给字体对话框 FontDialog1。

```
Fontdialog1.Font.Assign(Richedit1.DefAttributes);
```

2. 主要事件

(1) OnChange 事件：RichEdit 组件中的内容发生变化，触发该事件。

(2) OnSelectionChange 事件：选定文本触发该事件。

3. 主要方法

(1) CopyToClipboard 方法：复制到剪切板。

(2) CutToClipboard 方法：剪切到剪切板。

(3) PasteFromClipboard 方法：从剪切板粘贴，用剪切板上的内容覆盖选定的内容。

Lines 对象的主要方法有：

(1) SaveToFile 方法：将 RichEdit 组件中的内容保存。如下面的语句可以将 RichEdit1 中的内容保存到保存对话框 Savedialog1 所指定的文件名。

```
Richedit1.Lines.SaveToFile(Savedialog1.FileName);
```

(2) LoadFromFile 方法：在 RichEdit 中打开一个文件。如下面的语句可以将打开对话框 Opendialog1 所指定的文件在 RichEdit1 中打开。

```
Richedit1.Lines.LoadFromFile(Opendialog1.FileName);
```

4. RichEdit 组件的简单应用

【例 3-7】 编制一个简单的编辑器，要求该编辑器具有“复制”、“剪切”、“删除”、“粘贴”、“取消”、“全选”等功能。

分析：本程序将要用到 RichEdit 组件，并且要用到 RichEdit 组件的一些方法和属性。

程序设计步骤：

(1) 添加 RichEdit 组件 1 个、Button 组件 6 个到窗体，并调整大小和位置，界面如图 3-7 所示。

(2) 属性设置省略。

(3) 编写代码如下：

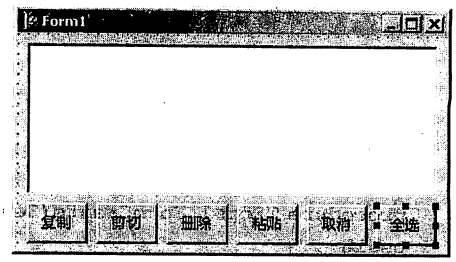


图 3-7　简单编辑器程序界面

```
procedure TForm1.Button1Click(Sender:
TObject);
begin
 richedit1.CopyToClipboard;                    //复制
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
 richedit1.CutToClipboard;                     //剪切
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
 richedit1.SelText:='';                        //删除
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
 richedit1.PasteFromClipboard;                 //粘贴
end;

procedure TForm1.Button5Click(Sender: TObject);
begin
 richedit1.Undo;                               //取消,仅对有些操作有效
end;

procedure TForm1.Button6Click(Sender: TObject);
begin
 richedit1.HideSelection:=false;               //选定文本反底显示
 richedit1.SelectAll;                          //选定所有文本
end;

procedure TForm1.RichEdit1SelectionChange(Sender: TObject);
begin
 if richedit1.SelLength=0
 then begin                                    //如果未做选定,则"复制"、"删除"、"剪切"按钮无效
      button1.Enabled:=false;
      button2.Enabled:=false;
      button3.Enabled:=false;
     end
 else begin                                    //如果选定了文本,则"复制"、"删除"、"剪切"按钮有效
```

```
        button1.Enabled:=true;
        button2.Enabled:=true;
        button3.Enabled:=true;
       end;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
 if richedit1.SelLength=0
 then begin                    //如果选定的文字为空,则"复制"、"删除"、"剪切"按钮无效
       button1.Enabled:=false;
       button2.Enabled:=false;
       button3.Enabled:=false;
      end
 else begin                    //如果选定的文字不为空,则"复制"、"删除"、"剪切"按钮有效
       button1.Enabled:=true;
       button2.Enabled:=true;
       button3.Enabled:=true;
     end;
end;
```

说明：虽然 RichEdit 没有提供删除方法，但是语句"richedit1. SelText :=";"的作用是将选定的文本置空，也就是删除选定的文本。这里的空字符千万不要写成空格字符，空字符和空格字符是有区别的。

【例 3-8】 请完善例 3-7 的功能，要求程序有"字体设置"、"打开"文件、"保存"文件、"新建"文件等功能。

分析：如果要真正地做一个编辑器，使用 Delphi 提供的向导是极其方便的。本程序的意义在于程序用到 RichEdit 组件提供的一些属性和方法，编写本程序有利于读者更加熟悉 RichEdit 组件的属性、方法和事件。

程序设计步骤：

(1) 设计界面，在例 3-7 界面的基础上添加 Button 按钮组件 4 个，分别是 Button7、Button8、Button9、Button10，添加打开对话框 OpenDialog1、保存对话框 SaveDialog1、字体对话框 FontDialog1。属性设置省略。

(2) 编写程序如下：

```
procedure TForm1.Button7Click(Sender: TObject);        //新建

 var i:word;

begin
 if richedit1.Modified                                 //如果 RichEdit1 中的内容被修改
 then begin
      i:=messagedlg('文件被修改,是否保
        存? ',mtconfirmation,[mbyes,mbno,mbcancel],1);
      if i=mryes then
       if savedialog1.Execute then
```

```
        begin                                          //保存之后再新建
         richedit1.Lines.SaveToFile(savedialog1.FileName);
         richedit1.Text:='';
         richedit1.Modified:=false;                    //新建之后,设置标志
        end;
       if i=mrno then
        begin                                          //不保存,直接新建
         richedit1.Text:='';
         richedit1.Modified:=false;                    //新建之后,设置标志
        end;
      end
 else begin                              //如果 RichEdit1 中的内容没有被修改,则直接新建
      richedit1.Text:='';
      richedit1.Modified:=false;                       //新建之后,设置标志
     end;
end;

procedure TForm1.Button8Click(Sender: TObject);        //打开

 var i:word;

begin
 if richedit1.Modified then                          //如果 RichEdit1 中的内容被修改
  begin
   i:=messagedlg('文件被修改,是否先保
        存?',mtconfirmation,[mbyes,mbno,mbcancel],1);
   if i=mryes then
    if savedialog1.Execute then
     begin                                           //先保存当前,再打开
      richedit1.Lines.SaveToFile(savedialog1.FileName);
      if opendialog1.Execute then
       richedit1.Lines.LoadFromFile(opendialog1.FileName);
      richedit1.Modified:=false;                     //打开之后,设置标志
     end;
    if i=mrno then
     if opendialog1.Execute then
      begin                                          //不保存修改,直接打开
       richedit1.Lines.LoadFromFile(opendialog1.FileName);
       richedit1.Modified:=false;                    //打开之后,设置标志
      end;
  end
 else                                  //如果 RichEdit1 中的内容没有被修改,则直接打开
  begin
   if opendialog1.Execute then
    richedit1.Lines.LoadFromFile(opendialog1.FileName);
   richedit1.Modified:=false;                        //打开之后,设置标志
  end;
end;
```

```
procedure TForm1.Button9Click(Sender: TObject);        //保存
begin
 if richedit1.Modified then                  //如果 RichEdit1 中的内容被修改则需要保存
  if savedialog1.Execute then
   begin
    richedit1.Lines.SaveToFile(savedialog1.FileName);
    richedit1.Modified:=false;                          //保存之后,设置标志
   end;
end;

procedure TForm1.Button10Click(Sender: TObject);       //字体
begin
 fontdialog1.Font.Assign(richedit1.DefAttributes);
                                  //打开对话框中的字体与 RichEdit 中的默认字体一样
 if fontdialog1.Execute then
  richedit1.SelAttributes.Assign(fontdialog1.Font);  //设置选中文本的字体
end;
```

说明：本程序用到了 RichEdit 的一些属性，如 Modified、Text、SelAttributes、Font、Lines 等，还用到了一些方法，如 SaveToFile、LoadFromFile 等。特别要注意的是在新建、打开和保存之后要将 Modified 属性值设置为 false，之后对 RichEdit 的任何修改都会再次将 Modified 的值改变为 true。

3.3 用于分组的组件

面板(Panel)和组框(GroupBox)可以用于将窗体上的其他组件进行分组。分组后的窗体会更加简洁明了。面板和组框这类组件又叫做“容器”类组件。

3.3.1 Panel 组件

Panel 面板是一个以多种三维效果显示的矩形区域，它可以将窗体分成规则的几块，为窗体中的组件提供可视化的分组。Panel 组件位于 Standard 选项卡中。

1. Panel 的主要属性

(1) Caption 属性：面板上显示的标题文字。

(2) Align 属性：Panel 的对齐方式、枚举类型有如表 3-10 所示的一些取值。

表 3-10 Align 属性取值及说明

属性值	说　明
alNone	组件的大小和位置不受 Parent 影响，可以放在 Parent 的任何位置，设置为任何大小
alTop	组件放置在 Parent 的顶边，组件 Width 属性没有意义，组件的宽度(Width)和 Parent 的宽度一致
alBottom	组件放置在 Parent 的底边，组件 Width 属性没有意义，组件的宽度(Width)和 Parent 的宽度一致

续表

属性值	说　明
alLeft	组件放置在 Parent 的左边，组件 Height 属性没有意义，组件的高度(Height)和 Parent 的高度一致
alRight	组件放置在 Parent 的右边，组件 Height 属性没有意义，组件的高度(Height)和 Parent 的高度一致
alClient	组件占据 Parent 的整个剩余空间，组件的 Width 属性和 Height 属性没有意义
alCustom	组件占据 Parent 的左上角，组件大小不会改变

(3) Anchors 属性：Panel 组件与 Parent 的大小和位置关系，说明组件与 Parent 的哪条边相连，其值是一个集合类型，如表 3-11 所示。

表 3-11　Anchors 属性取值及说明

集合元素	说　明	集合元素	说　明
akTop	到 Parent 顶边距离不变	akLeft	到 Parent 左边距离不变
akBottom	到 Parent 底边距离不变	akRight	到 Parent 右边距离不变

例如，如图 3-8 所示的 Panel1 的 Anchors 属性值为[akRight，akTop，akBottom]，则窗体变大的时候，Panel1 到窗体右边、上边、下边的距离均不变，但显然 Panel1 变高了。

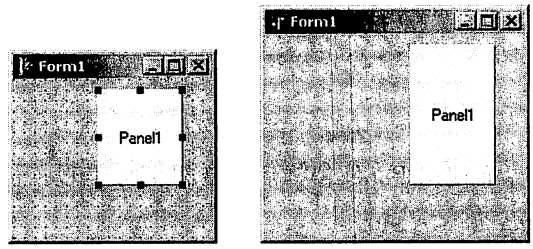


图 3-8　Panel1 的 Anchors 属性

要说明的是，Anchors 属性受 Align 属性的影响。例如，将 Panel1 的 Align 属性设置为 alLeft，显然此时如果 Anchors 的值含有集合元素 akTop 和 akBottom 就没有意义了。这点读者自己可以去体会。

2. Panel 组件的使用

【例 3-9】 Panel 组件的使用，界面如图 3-9 所示。

步骤如下：

(1) 在窗体上添加命令按钮 Button1，Caption 设置为“日期”，在窗体上添加 Panel1，设置 Panel1 的 Align 属性为默认值 alNone，并将 Panel1 的 BevelOuter 设置为 bvRaised。调整 Panel1 的位置和大小。

(2) 在 Panel1 中添加 Panel2，设置 Panel2 的 Align 属性为 alNone，并将 Panel2 的 BevelOuter 设置为 bvLowered。调整 Panel2 的位置和大小。

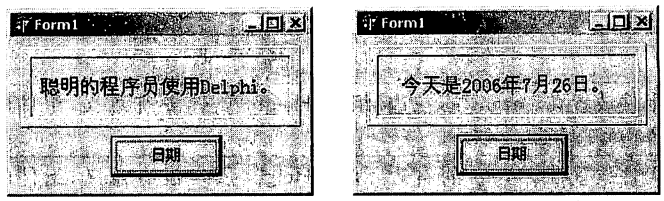


图 3-9 使用 Panel 修饰程序界面

(3) 编写程序如下：

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 Panel2.Font.Size:=12;
 panel2.Font.Name:='宋体';
 Panel2.Caption:='聪明的程序员使用 Delphi.';
end;

procedure TForm1.Button1Click(Sender: TObject);

 var year,month,day:word;

begin
 decodedate(date,year,month,day);
 //将 date 的年、月、日分别保存到其后的 3 个变量中
 panel2.Caption:='今天是'+inttostr(year)+'年'+inttostr(month)+'月'+inttostr
(day)+'日.';
end;
```

说明：很多程序的界面使用 Panel 修饰后会变得更加美观。

3.3.2 Splitter 组件

如果想建立一个在运行时可以自由改变区域大小的程序，这时可以使用 Splitter 组件。Splitter 组件虽然不是分组组件，但是它经常和 Panel 组件一起配合使用，它还经常和 Memo 组件、RichEdit 组件等配合使用。Splitter 组件位于 Additional 组件面板中。

1. Splitter 组件的主要属性

(1) Align 属性：属性值是枚举类型。值为 alLeft 或者 alRight 时，组件可以水平移动，控制左右两块区域改变大小，此时 Splitter 组件的 Height 属性无意义。值为 alTop 和 alBottom 时，组件可以垂直移动，控制垂直的两块区域改变大小，此时 Splitter 组件的 Width 属性无意义。Align 属性值一般不设置为 alNone 和 alClient。

(2) Beveled 属性：Boolean 型。值为 True 时，Splitter 组件呈凹形；默认值为 False 时，Splitter 组件呈凸形。

(3) Height 属性和 Width 属性：高度属性和宽度属性。当 Align 属性值为 alLeft 或者 alRight 时，Width 属性有效，可以设置分割条的宽度，此时 Height 属性无效，分割条的高度和 Parent 的高度相同。当 Align 属性值为 alTop 和 alBottom 时，Height 属性有效，

组件可以设置高度，此时 Width 属性无效，组件的宽度和 Parent 的宽度相同。

2. Splitter 组件和 Panel 组件的使用

【例 3-10】 Splitter 组件和 Panel 组件的使用。

步骤如下：

(1) 新建 Application，在窗体中添加 Memo1，设置它的 Align 属性为 alLeft，这样 Memo1 位于窗体的左边。

(2) 在窗体中添加 Splitter1，设置其 Align 属性为 alLeft，这样 Splitter 就紧贴左边 Memo1 组件，设置 Splitter1 的 Width 为 2。

(3) 在右边空白处添加 Panel1，设置 Panel1 的 Align 属性为 alClient，Panel1 就占据了右边剩下的所有区域。

(4) 在 Panel1 中添加 Memo2，设置它的 Align 属性为 alBottom，这样 Memo2 就占据窗体的右下角(Panel1 的下面)。

(5) 在 Panel1 中添加 Splitter2，设置 Splitter2 的 Align 属性为 alBottom，这样 Splitter2 就紧贴在 Memo2 的上面，并设置 Splitter2 的 Height 属性为 2。

(6) 在 Panel1 中添加 Memo3，设置 Memo3 的 Align 属性为 alClient。这样 Memo3 就占据了 Panel1 剩余的空间。

(7) 设置 Memo1、Memo2 和 Memo3 的 Lines 属性值，显示如图 3-10 所示文字。

程序运行，界面如图 3-10 所示，可以通过移动 Splitter 来改变 3 个 Memo 区域的大小。

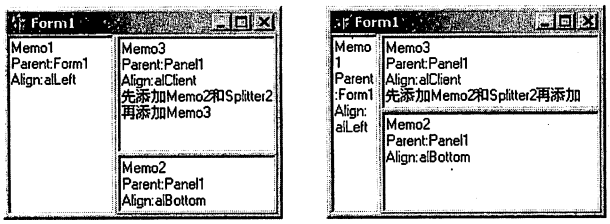


图 3-10　利用 Splitter 和 Panel 分割界面

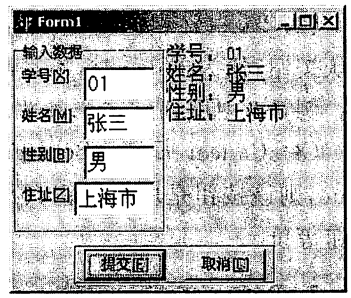


图 3-11　程序界面

3.3.3　GroupBox 组件

组框 GroupBox 是又一个用于分组的组件，它类似于面板 Panel，但是其 Caption 显示在边框上，这点与面板 Panel 有些不同。GroupBox 不仅可以用来美观，还可以用于不同组的组件之间的协调操作，因此 GroupBox 在单选框和复选框中用得很多(将在后面章节中讲解)。GroupBox 组件位于 Standard 组件面板上。

【例 3-11】 利用 GroupBox 将例 3-5 界面做一些处理，界面如图 3-11 所示。

步骤如下：

(1) 同时选定 Label1、Label2、Label3、Label4、Edit1、Edit2、Edit3 和 Edit4，并剪切这 8 个组件。

(2) 在窗体的左边添加一个 GroupBox1,调整好位置大小,设置 Caption 为“输入数据”。

(3) 将步骤(1)中选定的 8 个组件粘贴到 GroupBox1 中。

(4) 同时选定 Button1 和 Button2,并剪切这两个组件。

(5) 在窗体下面添加组件 GroupBox2,并设置 Caption 为空。

(6) 将步骤(4)中剪切的 Button1 和 Button2 粘贴到 GroupBox2 中。

(7) 调整这些组件的位置和大小。

说明:使用 GroupBox 的目的是使得程序界面更加美观,有时候可能是分组需要(单选按钮的分组)。

3.4 按钮类组件

按钮是使用最为广泛的组件,Delphi 提供的按钮有普通按钮 Button、位图按钮 BitBtn、加速按钮 SpeedButton、复选按钮 CheckBox(也叫复选框)、单选按钮 RadioButton 和单选按钮组 RadioGroup。

3.4.1 Button 组件

Button 组件是最常见的组件之一,位于 Standard 组件面板中。

1. Button 的主要属性

(1) Caption 属性:组件的标题文字,可以使用符号“&”来设置按钮的加速键,前面已经使用到。

(2) Default 属性:Boolean 型,决定按钮是否为默认按钮。如果值为 True,则按钮是默认按钮;值为 False 时,不是默认按钮。如果是默认按钮,单击按钮和按回车键是等价的。例 3-5 中用到过这个属性。

(3) Cancel 属性:Boolean 型,决定按钮是否为取消按钮。如果 Cancel 属性值为 True,则该按钮是取消按钮;否则不是取消按钮。如果是取消按钮,则按 Esc 键和单击该按钮等价。

2. Button 的主要事件

Button 的主要事件有 OnClick 等事件,下列情形之一会触发 OnClick 事件。

(1) 单击按钮。

(2) 按钮获得焦点的时候按回车键或者空格键。

(3) 执行语句 Button 对象. Click。

3.4.2 BitBtn 组件

BitBtn 组件也是最常见的组件之一,位于 Additional 组件面板中。该组件可以在按钮中显示一幅格式为 bmp 的图片,以使按钮更加美观和形象。

1. BitBtn 组件的主要属性

除了有 Caption、Default、Cancel 等属性之外,BitBtn 还有如下一些主要属性。

(1) Glyph 属性:为按钮指定一个图标文件(扩展名为 bmp),该图标显示在按钮表

面，Tbitmap 型。

(2) Kind 属性：枚举型，决定按钮的类型。取值有 bkCustom、bkOK、bkCancel、bkHelp、bkYes、bkNo、bkClose、bkAbort、bkRetry、bkIgnore、bkAll。默认值为 bkCustom，表示种类是自定义类型，位图由 Glyph 决定。其他值分别对应不同的位图，并将 ModalResult 自动设置为相应的值。如 bkOK 对应 mrOK，bkYes 对应 mrYes 等。

2. BitBtn 按钮的使用

【例 3-12】 在窗体上添加标签 Label1，位图按钮 BitBtn1～BitBtn11。分别设置它们的 Kind 属性，设置 BitBtn5 的 Caption 为“保存”，Kind 为 bkCustom，设置 Glyph 属性为一个图标。界面如图 3-12 所示。

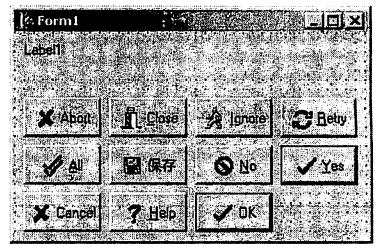


图 3-12　各种 Kind 类型的 BitBtn 按钮

编写程序：

```
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
 label1.Caption:='你按了 Abort 按钮';
end;
```

或者：

```
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
 if ModalResult=mrAbort          //返回 mrAbort 表示按了 Abort 类型的按钮
 then label1.Caption:='你按了 Abort 按钮';
end;
```

说明：其他按钮的代码类似，要注意的是 BitBtn4 按钮(Close 按钮)不需要编写代码，单击它即可关闭程序。

3.4.3　SpeedButton 组件

SpeedButton 加速按钮组件也是最常见的组件之一，位于 Additional 组件面板中。使用它可以设计若干组互斥的按钮选项，只需要将多个 SpeedButton 的 GroupIndex 属性设置为相同且不为 0，则这些按钮会成为一组，它们每次仅可以按下一个。SpeedButton 不接受输入焦点，没有切换次序。

1. SpeedButton 的特殊属性

SpeedButton 的主要属性与 Button 类似，特殊属性如下。

(1) Down 属性：按钮处于按下状态，Down 值为 True；否则 Down 值为 False。

(2) Flat 属性：Boolean 型。当值为 False 时，外观是立体按钮；值为 True 时，外观是平面按钮，外观和标签一样，只有鼠标移上去的时候才显示为立体。

(3) GroupIndex 属性：若值大于 0，则相同值的若干个 SpeedButton 按钮协同工作，任何时刻这些按钮至多只能有一个被按下。值为 0 表示该按钮不与其他按钮协同工作。

(4) Glyph 属性：按钮上显示的图片，与 BitBtn 类似。

2. SpeedButton 按钮的应用

【例 3-13】 利用 SpeedButton 按钮设置字体应用程序，界面如图 3-13 所示。

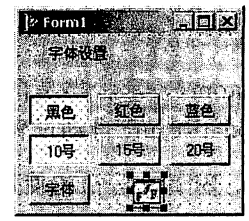


图 3-13 SpeedButton 按钮举例

步骤如下：

(1) 在窗体中添加组件 Label1、SpeedButton1～SpeedButton7、FontDialog1 等组件。调整这些组件的大小和位置。

(2) 设置这些组件的属性，参见表 3-12。

表 3-12 SpeedButton 按钮的属性设置表

对 象	属 性	属 性 值	说 明
SpeedButton1	Caption	黑色	标题
	Down	True	按下状态
	GroupIndex	1	SpeedButton1～SpeedButton3 是一组
SpeedButton2	Caption	红色	标题
	Down	False	
	GroupIndex	1	SpeedButton1～SpeedButton3 是一组
SpeedButton3	Caption	蓝色	标题
	Down	False	
	GroupIndex	1	SpeedButton1～SpeedButton3 是一组
SpeedButton4	Caption	10 号	标题
	Down	True	按下状态
	GroupIndex	2	SpeedButton4～SpeedButton6 是一组
SpeedButton5	Caption	15 号	标题
	Down	False	
	GroupIndex	2	SpeedButton4～SpeedButton6 是一组
SpeedButton6	Caption	20 号	标题
	Down	False	
	GroupIndex	2	SpeedButton4～SpeedButton6 是一组
SpeedButton7	Caption	字体	标题
	Down	False	
	GroupIndex	0	不分组

(3) 本题可以将 SpeedButton1～SpeedButton3 的 OnClick 事件过程写在一起。方法是：同时选定 SpeedButton1～SpeedButton3，然后在 Object Inspector 的 Events 选项卡

中选定 OnClick 事件，双击其右边的空格显示 SpeedButton1Click（表示单击 SpeedButton1～SpeedButton3 都会执行 TForm1. SpeedButton1Click 过程），并进入代码编辑器。编写如下代码：

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
//单击 SpeedButton1～SpeedButton3 都会执行 TForm1.SpeedButton1Click 过程
//此处的代码也可以写成 3 个过程
begin
 if (sender as tspeedbutton).caption='黑色'       //单击了“黑色”
 then label1.Font.Color:=rgb(0,0,0);
 if (sender as tspeedbutton).caption='红色'       //单击了“红色”
 then label1.Font.Color:=rgb(255,0,0);
 if (sender as tspeedbutton).caption='蓝色'       //单击了“蓝色”
 then label1.Font.Color:=rgb(0,0,255);
end;
```

（4）继续编写 SpeedButton4～SpeedButton7 的代码，如下：

```
procedure TForm1.SpeedButton4Click(Sender: TObject);
begin
 label1.Font.Size:=10;
end;

procedure TForm1.SpeedButton5Click(Sender: TObject);
begin
 label1.Font.Size:=15;
end;

procedure TForm1.SpeedButton6Click(Sender: TObject);
begin
 label1.Font.Size:=20;
end;

procedure TForm1.SpeedButton7Click(Sender: TObject);
begin
 fontdialog1.Font.Assign(label1.Font);
                    //字体对话框初始字体与标签的字体一样
 if fontdialog1.Execute
 then label1.Font.Assign(fontdialog1.Font)
                    //单击字体对话框的“确定”按钮后，标签取字体对话框的字体
end;
```

SpeedButton4～SpeedButton6 的程序代码也可以写成一个过程，方法和 SpeedButton1～SpeedButton3 类似。

（5）运行结果如图 3-14 所示。

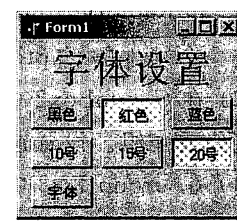


图 3-14　SpeedButton 程序举例

说明：本处将颜色按钮分成一组，将字号大小按钮分成一组，而字体不需要分组。这样分组后就可以同时选择颜色和大小了。

3.4.4 CheckBox 组件

CheckBox 组件是复选按钮组件，它是一个带标签的方框(☐)，☑表示被选中，☐表示未被选中。选中时 Checked 属性值为 True，未被选中时 Checked 属性值为 False，默认值为 False。多个复选按钮之间互不影响。CheckBox 组件在 Standard 组件面板中。

CheckBox 组件的事件有 OnClick 事件，下列情况下发生 OnClick 事件。

(1) 单击 CheckBox 组件(Checked 属性的值也会自动改变)。

(2) 用代码改变 Checked 属性的值。

CheckBox 组件的特点是每次单击都会改变 Checked 属性的值，并触发 OnClick 事件。

3.4.5 RadioButton 组件

RadioButton 组件是单选按钮组件，它可以用于显示一组互斥的选项。同一组单选按钮中最多只能有一个单选按钮被选中，一旦选定某个单选按钮，则此前被选定的那个单选按钮变为未选定状态。如果想多选，可以配合 GroupBox 组件来实现，可以把 GroupBox 中的多个 RadioButton 按钮看成一组，有几个 GroupBox 就将 RadioButton 分成几组。RadioButton 按钮在 Standard 组件面板中。

RadioButton 的 Checked 属性用于表示单选按钮是否被选中。◉表示被选中，Checked 属性值为 True；○表示未被选中，Checked 属性值为 False；默认值为 False。

RaidoButton 的事件主要有 OnClick 事件，下列情况下触发 OnClick 事件。

(1) 单击 RadioButton 组件使得其 Checked 属性值从 False 变为 True。

(2) 通过代码使得其 Checked 属性值从 False 变为 True。

RadioButton 组件的特点是只有单击未被选中的按钮才会触发 OnClick 事件，并将 RadioButton 的 Checked 属性值从 False 改变为 True。

【例 3-14】 利用 CheckBox 和 RadioButton 组件编写字体设置程序，界面如图 3-15 所示。

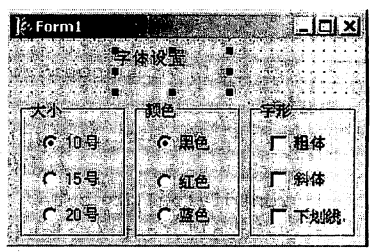


图 3-15 字体设置程序界面

步骤如下：

(1) 在窗体中添加 Label1、GroupBox1、GroupBox2、GroupBox3。设置 Caption 属性分别为“字体设置”、“大小”、“颜色”和“字形”，调整好这 4 个组件的大小和位置。

(2) 在 GroupBox1 中添加 RadioButton1、RadioButton2 和 RadioButton3，此 3 个单选按钮分成一组。设置这 3 个组件的 Caption 分别为“10 号”、“15 号”和“20 号”。

(3) 在 GroupBox2 中添加 RadioButton4、RadioButton5 和 RadioButton6，此 3 个单选按钮分成一组。设置这 3 个组件的 Caption 分别为“黑色”、“红色”和“蓝色”。

(4) 在 GroupBox3 中添加 CheckBox1、CheckBox2 和 CheckBox3，设置这 3 个复选按钮的 Caption 分别为“粗体”、“斜体”和“下划线”。

说明：GroupBox1 和 GroupBox2 的作用是分组，使 RadioButton1、RadioButton2 和

RadioButton3 成为一组，而 RadioButton4、RadioButton5 和 RadioButton6 也成为一组，这样就可以提供多选了。GroupBox3 的作用是美观界面，复选按钮并不需要分组协调工作。

(5) 编写程序：

```
procedure TForm1.FormCreate(Sender: TObject);
begin
                                          //程序运行最开始字号是 10 号，颜色是黑色
 radiobutton1.Checked:=true;
 radiobutton4.Checked:=true;
end;

procedure TForm1.RadioButton1Click(Sender: TObject);
begin
 label1.Font.Size:=10;
end;

procedure TForm1.RadioButton2Click(Sender: TObject);
begin
 label1.Font.Size:=15;
end;

procedure TForm1.RadioButton3Click(Sender: TObject);
begin
 label1.Font.Size:=20;
end;

procedure TForm1.RadioButton4Click(Sender: TObject);
begin
 label1.Font.Color:=rgb(0,0,0)
end;

procedure TForm1.RadioButton5Click(Sender: TObject);
begin
 label1.Font.Color:=rgb(255,0,0)
end;

procedure TForm1.RadioButton6Click(Sender: TObject);
begin
 label1.Font.Color:=rgb(0,0,255)
end;

procedure TForm1.CheckBox1Click(Sender: TObject);
begin
 if CheckBox1.Checked
 then label1.Font.Style:=label1.Font.Style+[fsbold]
 else label1.Font.Style:=label1.Font.Style-[fsbold]
end;

procedure TForm1.CheckBox2Click(Sender: TObject);
```

```
begin
 if CheckBox2.Checked
 then label1.Font.Style:=label1.Font.Style+[fsitalic]
 else label1.Font.Style:=label1.Font.Style-[fsitalic]
end;

procedure TForm1.CheckBox3Click(Sender: TObject);
begin
 if CheckBox3.Checked
 then label1.Font.Style:=label1.Font.Style+[fsunderline]
 else label1.Font.Style:=label1.Font.Style-[fsunderline]
end;
```

分析：单选按钮的 OnClick 事件过程中不需要使用 if 语句，而复选按钮的 OnClick 事件过程中需要使用 if 语句，这与二者的 OnClick 事件触发的规律有关系。这里用到了集合的并运算，应该不难理解。

3.4.6 RadioGroup 组件

RadioGroup 组件是类似于多个 RadioButton 和一个 GroupBox 组件的结合，位于 Standard 组件面板中。

RadioGroup 组件的主要属性有：

(1) Colums 属性：指明 RadioGroup 的列数，取值范围是 1～16，默认值是 1，类型是 Integer。

(2) Items 属性：每个单选项旁边的文字提示，String 类型数组，可以使用类似 RadioGroup1. Items[i]来表示各个选项的标题，i=0 表示第一项。

(3) ItemIndex 属性：指示当前选中项的序号，类型是 Integer，序号从 0 开始。默认值为－1，表示不选中任何项。

关于 RadioGroup，图 3-16 示意了其最重要的几个属性。

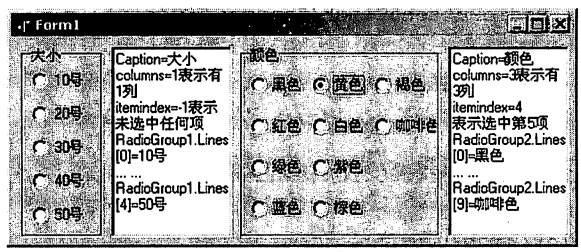


图 3-16 RadioGroup 的某些属性

3.5 列表框和组合框

列表框(ListBox)和组合框(ComboBox)可以为用户提供多项选择，组合框还允许用户通过键盘输入数据，相当于编辑框 Edit 与列表框(ListBox)的结合。列表框和组合框在 Standard 组件面板中。

3.5.1　ListBox 组件

ListBox 组件以列表的形式显示多项供用户选择。如果不能显示全部项，还可以添加滚动条使用户通过上下滚动条来查阅所有选项。

1. ListBox 组件的主要属性

(1) Items 属性：列表选项的集合，String 类型数组，如 ListBox1. Items[3]表示 ListBox1 的第 4 项。关于 Items 有如下一些用法：

```
ListBox1.Items.Add('添加一行');              //在最后添加一项
ListBox1.Items.Delete(5);                    //删除第 5 项
ListBox1.Items.Insert(3,'插入一行');         //在第 4 行之前插入一行
ListBox1.Items.Move(3,5);                    //将第 4 行移动到第 6 行
St1:=ListBox1.Items[3];                      //将第 4 行的字符赋值给字符变量 st1
Memo1.Lines:=Listbox1.Items;                 //将 Listbox1 中的所有项显示在 Memo1 中
n:=ListBox1.Items.count;                     //n 为 ListBox1 的项数
ListBox1.Items.Clear;                        //清空 Listbox1 中的所有项
```

(2) ItemsIndex 属性：选定项的索引值，值为 i 时表示选定了第 i+1 项，值为 -1 表示未选定任何项。例如：

```
ListBox1.Items[ListBox1.ItemsIndex];         //当前选中项的字符串
ListBox1.ItemIndex:=3;                       //选中第 4 项
```

(3) Sorted 属性：Boolean 型，值为 True 时表示选项排序，值为 False 时表示选项不排序。

(4) Columns 属性：Integer 型，列表的列数。

(5) MultiSelect 属性：Boolean 型，是否允许选择多项。

(6) Selected 属性：用来设置或者返回某项是否被选中了。

关于 Selected 属性的用法如下。

```
listbox1.Selected[2]:=true;          //选中第 3 项
if listbox1.Selected[2] then begin...end;
                                     //如果第 3 项被选中就执行 begin 和 end 间的语句
```

2. ListBox 组件的主要方法

Clear 方法：清除 ListBox 中的所有项数。下面两种方法均可。

```
ListBox1.Items.Clear;
ListBox1.clear;
```

3. ListBox 组件的事件

(1) OnClick 事件：选择某项时触发该事件。

(2) OnEnter 事件：得到焦点时触发该事件。

(3) OnExit 事件：失去焦点时触发该事件。

4. ListBox 组件的应用

【例 3-15】 在列表框中选择自己喜欢的城市，界面如图 3-17 所示。

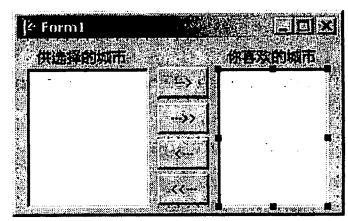


图 3-17　ListBox 程序界面

步骤如下：

(1) 在窗体中添加标签 Label1、Label2、命令按钮 Button1～Button4(从上到下)，添加两个列表框 ListBox1(左边)和 ListBox2(右边)。

(2) 设置这些组件的属性，参见表 3-13。

表 3-13 ListBox 的属性设置

对 象	属 性	属性值	说 明
ListBox1	MultiSelect	True	可以选择多项
	Sorted	True	选项排序
ListBox2	MultiSelect	True	可以选择多项
	Sorted	True	选项排序

(3) 编写程序，如下：

```
procedure TForm1.Button1Click(Sender: TObject);

 var i:integer;

begin
 i:=0;          //从第一项开始
 while i<listbox1.Items.Count do
                //直到最后一项
  if listbox1.Selected[i] then
                //判断每一项是否被选中,若被选中则执行 begin 和 end 间的语句
   begin
    listbox2.Items.Add(listbox1.Items[i]);
                //将选中的项添加到 listbox2
    listbox1.Items.Delete(i);
                //并从 listbox1 中删除
   end
  else
   i:=i+1;      //测试下一项
end;
```

说明：如果某个城市没有被选中，则 i+1 测试下一个城市；如果某个城市被选中，则从左边列表框中删除该城市。细心的读者会注意到删除某个城市(项)后 i 并没有加 1，原因是相同的 i 值，在完成删除操作之前和之后是表示不同的项的。

继续编写下面的代码：

```
procedure TForm1.Button2Click(Sender: TObject);

 var i:integer;

begin
 for i:=0 to listbox1.Items.Count-1 do
  listbox2.Items.Add(listbox1.Items[i]);
```

```
  //将 listbox1 中所有项添加到右边 listbox2
  listbox1.Items.Clear;
  //清除 listbox1 中所有项
end;

procedure TForm1.Button3Click(Sender: TObject);

 var i:integer;

begin
 i:=0;
 while i<listbox2.Items.Count do
   if listbox2.Selected[i] then
    begin
     listbox1.Items.Add(listbox2.Items[i]);
     listbox2.Items.Delete(i);
    end
  else
   i:=i+1;
end;

procedure TForm1.Button4Click(Sender: TObject);
 var i:integer;
begin
 for i:=0 to listbox2.Items.Count-1 do
  listbox1.Items.Add(listbox2.Items[i]);
 listbox2.Items.Clear;
end;
```

3.5.2　ComboBox 组件

ComboBox 组件是组合框组件，兼有 Edit 组件和 ListBox 组件的功能，用户可以直接输入文本或者在列表中选择选项。ComboBox 不仅具有很多和 ListBox 相同的属性，还具有自己独特的属性。

1. ComboBox 组件的特殊属性

(1) DropDownCount 属性：下拉部分可以显示的项数，如果不能显示全部项数，可以通过滚动条来显示其他选项。默认值是 8。

(2) SelText 属性：编辑区选择的文本。

(3) SelLength 属性：编辑区选定文本的长度。

(4) SelStart 属性：编辑区选定文字的起点，编辑区第一个字符位置是 0。

(5) Style 属性：ComboBox 的风格。常见的取值如下。

csDropDown：可编辑且有下拉列表。

csDropDownList：编辑框不可编辑，有下拉列表框。

CsSingle：编辑框可以编辑，下拉列表框直接显示出来(不用下拉)。

(6) Text 属性：ComboBox 中的文本。

2. ComboBox 组件的方法

ComboBox 组件具有 ListBox 组件和 Edit 组件的方法。

(1) Clear 方法：与 ListBox 有一些差别，如：

```
ComboBox1.Clear;          //清除 ComboBox1 中的编辑框和列表框中所有内容
ComboBox1.Items.Clear;    //清除列表项中所有内容
```

(2) SelectAll 方法：选定编辑框中所有内容。

(3) SetFocus 方法：ComboBox 得到焦点。

ComboBox 组件还具备 Edit 和 ListBox 组件的其他一些方法，在此不再赘述。

3. ComboBox 组件的事件

ComboBox 组件具备 Edit 组件和 ListBox 组件的一些事件。如：

(1) OnClick 事件：鼠标单击事件，鼠标单击选择某项。

(2) OnChange 事件：编辑框内容被改变。

(3) OnKeyPress 事件：按键事件，参数与 Edit 的按键事件相同。

(4) OnKeyUp 事件和 OnKeyDown 事件：类似 Edit 的相应事件。

(5) OnDropDown 事件：当用户单击右边箭头按钮即打开下拉列表，此时触发该事件。

4. ComboBox 组件的应用

【例 3-16】 字体设置程序，窗体上有编辑框 Edit1，要求通过 ListBox1 和 ComboBox1 来设置其字体和大小，界面如图 3-18 所示。请编写程序完成。

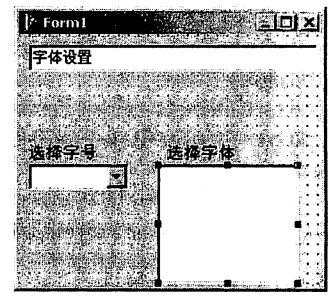


图 3-18 字体设置界面

分析：我们希望编写一个质量较高的程序，如少出错、界面友好、美观、操作方便等。为了操作方便，本程序不仅可以选择字号，还允许用户直接输入字号。在防止出错方面，程序要求用户在输入字号的时候禁止非数字字符的输入。另外，还禁止设置 40 号以上的字号(太大，影响美观)。本程序用到 OnKeyPress、OnClick 等事件。

步骤如下：

(1) 在窗体上添加组件 Edit1、Label1、Label2、ComboBox1 和 ListBox1。简单设置属性，并调整组件的大小和位置。

(2) 根据上面的分析，编写如下代码：

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 listbox1.Items:=screen.Fonts;          //在 ListBox1 中显示字体
 combobox1.Items.Add('8');              //添加字号
 combobox1.Items.Add('12');
 combobox1.Items.Add('16');
 combobox1.Items.Add('20');
 combobox1.Items.Add('24');
```

```
 combobox1.Items.Add('28');
 combobox1.Items.Add('32');
 combobox1.Items.Add('36');
 combobox1.Items.Add('40');                    //最大字号 40 号
 combobox1.Text :=combobox1.Items[0];          //ComboBox1 编辑框中显示 8 号字
end;

procedure TForm1.ComboBox1KeyPress(Sender: TObject; var Key: Char);
begin
//在编辑框中输入字号
 if not ((key>='0') and (key<='9') or (key=#8) or (key=#13))
 then key:=#0;                                 //key:=#0 表示什么也不错,#0 是空字符
 //禁止'0'~'9'、BackSpace 和回车键之外的键输入
 if (length(combobox1.Text)>0)and(key=#13) and (strtoint(combobox1.Text)<=
40) then
 edit1.Font.Size:=strtoint(combobox1.Text)
 //编辑框不为空,同时输入的数字小于或者等于 40,并且按下回车键
end;

procedure TForm1.ComboBox1Click(Sender: TObject);
begin
 edit1.Font.Size:=strtoint(combobox1.Text)
 //直接选择字号
end;
procedure TForm1.ListBox1Click(Sender: TObject);
begin
 edit1.Font.Name:=listbox1.Items[listbox1.ItemIndex];
 //选择字体,listbox1.Items[listbox1.ItemIndex]为用户在列表框中选择的项
end;
```

(3) 运行程序,界面如图 3-19 所示。

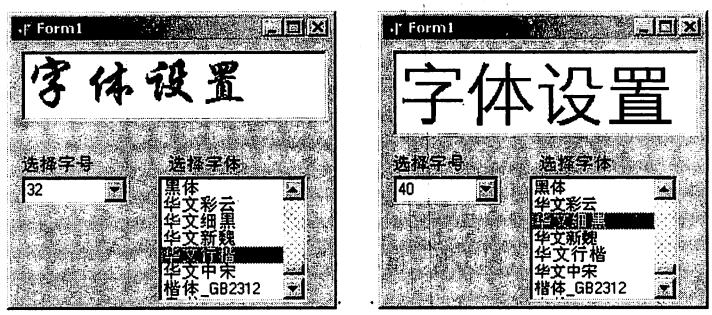


图 3-19　字体设置运行界面

说明：本程序 TForm1.ComboBox1KeyPress 中的代码非常实用,在最大程度上防止了错误的出现。因为输入非数字字符会引起错误,即使空白也会出错,本程序可以防止这些情况的发生。本程序的巧妙之处还在于允许按 BackSpace 键,提供了输入错误后的修改可能。

3.6 计时器

计时器组件(Timer)是一个非可视化的组件,它能定时触发 OnTimer 事件。如果想定时执行某些语句,可以考虑使用 Timer 组件。Timer 组件在 System 组件面板上。

1. Timer 组件的属性

(1) Enabled 属性:当 Enabled 属性值为 True 时,打开定时器;当值为 False 时,Timer 组件关闭计时器。默认值为 True。

(2) Interval 属性:控制 OnTimer 事件发生的时间间隔,单位是毫秒,类型是 Integer。将 Interval 属性值设置为 0 时,相当于关闭计时器。Interval 的默认值是 1000,即 1 秒钟。

2. Timer 组件的事件

OnTimer 事件:当 Timer 组件有效时,它每隔 Interval 毫秒触发 OnTimer 事件,执行相应的程序。

3. Timer 组件的程序举例

【例 3-17】 设置一个滚动字幕,让字符从下向上滚动。

分析:为了控制字符向上滚动,显然要使用计时器来控制字符滚动的快慢。并设置其 Interval 用于控制时间间隔,时间间隔越短,字幕滚动越快。

步骤如下:

(1) 添加组件 Timer1、Label1,并设置字号为 12 号,设置字体为"宋体"。调整好 Label1 的大小和位置,设置 Timer1 的 Interval 为 10。界面如图 3-20 所示。

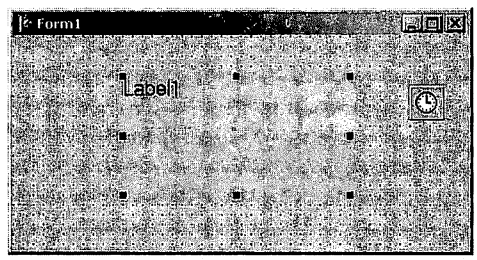


图 3-20 滚动字幕程序界面

(2) 根据分析,编写代码如下:

在 TForm1.FormCreate 中编写代码,实现程序运行一开始就显示李白的"静夜思"的功能。

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 label1.Caption:='静夜思'+chr(13);
 label1.Caption:=label1.Caption+'床前明月光,'+chr(13);
 label1.Caption:=label1.Caption+'疑是地上霜;'+chr(13);
```

```
 label1.Caption:=label1.Caption+'举头望明月,'+chr(13);
 label1.Caption:=label1.Caption+'低头思故乡。'
end;
```

chr(13)表示换行,在 Timer1Timer 事件过程中编写代码实现字幕滚动功能。

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
 if label1.Height+label1.Top<=0          //字幕滚出 Form1
 then label1.Top:=form1.Height           //字幕重新从下方显示出来
 else label1.top:=label1.top-1;          //字幕向上移动
end;
```

说明:程序可以通过设置 Interval 属性值来控制字幕滚动快慢,或者通过设置语句"label1. top :=label1. top－x;"来控制字幕滚动快慢,x 越大字幕滚动越快。

【例 3-18】 在例 3-17 基础上添加一个按钮,用于控制"继续"和"暂停",请编写代码完成该功能。

分析:添加一个按钮 Button1,Button1 的初始 Caption 显然应该设置为"暂停(&S)",因为程序运行最开始,字幕是处于滚动状态的;当用户单击"暂停(&S)"按钮时,滚动字幕应暂停,"暂停(&S)"按钮变成"继续(&C)"按钮;单击"继续(&C)"按钮,按钮变成"暂停(&S)",并且字幕开始滚动,如此周而复始。

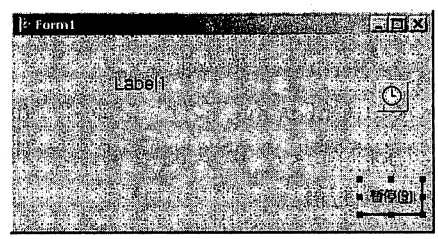


图 3-21　添加"暂停"按钮后的界面

(1) 添加 Button1 按钮,设置 Caption 属性为"暂停(&S)",界面如图 3-21 所示。

(2) 为 Button1 编写 OnClick 事件过程,如下:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 if button1.Caption='暂停(&S)'then
  begin
   button1.Caption:='继续(&C)';
   timer1.Enabled:=false;
  end
 else
  begin
   button1.Caption:='暂停(&S)';
   timer1.Enabled:=true;
  end;
end;
```

说明:本程序还可以进一步改进,比如在窗体上添加一个编辑框,用于输入 Timer 的时间间隔,以便于控制滚动快慢。

3.7 滚动条组件

如果窗体或者其他组件中的内容一次显示不完，可以考虑使用滚动条组件 ScrollBar，滚动条还可以用来改变一些量。滚动条是一个常用组件，在 Standard 组件面板中。

1. ScrollBar 组件的属性

(1) Kind 属性：用于决定滚动条是水平放置还是垂直放置，其值有 sbHorizontal 和 sbVertical 之分，前者表示滚动条水平放置，后者表示滚动条垂直放置。

(2) Max 属性和 Min 属性：用来决定滚动条表示数值的范围，即滚动条的最大值和最小值。

(3) Position 属性：滚动条的值，也就是滚动条滑块的位置，它在 Min 和 Max 之间。

(4) SmallChange 属性：鼠标单击滚动条两边的箭头，滚动条每次移动的距离。

(5) LargeChange 属性：鼠标在滚动条(但不是滑块)上单击一次，滑块滚动的距离。按 PgDn 或者 PgUp 滚动条改变的量也是 LargeChange。

2. ScrollBar 组件的事件

(1) OnChange 事件：滚动条 Position 值的改变就会触发 OnChange 事件，比如拖动滚动条的箭头或者单击滚动条滑块和滚动条空白处。也可以使用代码改变滚动条 Position 的值。

(2) OnScroll 事件：使用鼠标操作改变滚动条 Position 的值触发该事件，此时 OnChange 事件也会发生。但是，通过代码改变 Position 的值是不会触发 OnScroll 事件的。

3. ScrollBar 组件的应用

【例 3-19】 使用滚动条来设置窗体颜色。

分析：可以使用 3 个滚动条分别来表示红色、绿色、蓝色，改变滚动条的值时会触发 OnChange 事件，在 OnChange 事件过程中编写代码改变窗体的颜色。

步骤如下：

(1) 在窗体上添加 3 个标签 Label1～Label3 和 3 个滚动条 ScrollBar1～ScrollBar3。设置 3 个标签的 Caption 分别为“红”、“绿”和“蓝”。设置 3 个滚动条的 Min 均为 0，Max 均为 255，LargeChange 和 SmallChange 均为 1，Kind 均为 sbHorizontal。调整这些组件的位置和大小。程序界面如图 3-22 中的左图所示。

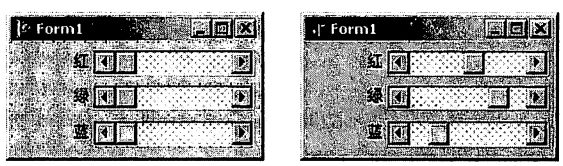


图 3-22 利用滚动条改变窗体颜色

(2) 编写程序如下：

```
procedure TForm1.yanse();
begin
 form1.Color:=rgb(scrollbar1.Position,scrollbar2.Position,scrollbar3.Position)
end;

procedure TForm1.ScrollBar1Change(Sender: TObject);
begin
 yanse
end;

procedure TForm1.ScrollBar2Change(Sender: TObject);
begin
   yanse
end;

procedure TForm1.ScrollBar3Change(Sender: TObject);
begin
  yanse
end;
```

(3) 程序运行，如图 3-22 中的右图所示。

说明：本处因改变 ScrollBar1、ScrollBar2 和 ScrollBar3 的值都会改变窗体颜色，因此首先定义一个过程 TForm1. yanse()用于改变颜色，然后在后面每一个 ScrollBar 的 OnChange 事件过程中再来调用该过程。也可以将改变颜色的语句直接写进每个 ScrollBar 的 OnChange 事件过程，不要忘记在前面声明过程 procedure yanse。

3.8　多选项卡组件

有时候窗体上的内容太多，一个窗体无法显示出来，此时可以考虑使用多个窗体，但是在每个窗体之间切换非常不方便。Delphi 提供了多选项卡组件 PageControl，该组件可以实现一个窗体上浏览多个页面(称之为选项卡)的功能。

该组件位于 Win32 组件面板上，适合每个选项显示一个独立的信息，该组件的每页都是一个 TabSheet。

1. PageControl 组件的属性

(1) Style 属性：选项卡的样式。

(2) Images 属性：指定从哪个图像列表选择选项卡的图标。

(3) MultiSelect 属性：可否选择多个选项页面。值为 True 时可以选定多页，值为 False 时不能够选定多页。

(4) ActivePage 属性：显示当前选中的选项卡，用于切换选项卡。

(5) PageCount 属性：选项卡有多少页面，只读属性。

(6) Pages 属性：数组，Pages[n]，表示第 n+1 个页面，n=0 表示第一个页面。

例如：

```
Label1.Caption:=Pagecontrol1.Pages[2].Caption;
```

表示在标签上显示第 3 页页面的标题文字。

组件的每一页是一个 TabSheet 组件，该组件的属性如下：

(1) ImageIndex 属性：该选项卡在 ImageList 中选择的图标的序号。

(2) TabVisible 属性：控制或者返回该选项卡是否显示。

(3) PageIndex 属性：该选项卡在 PageControl 中的索引号。

(4) TabIndex 属性：该选项卡在所有可见选项卡中的序号。

(5) PageControl 属性：返回该选项卡所在的 PageControl。

下面介绍如何使用 PageControl 组件。在窗体上添加一个 PageControl 组件如 PageControl1。选择该组件，单击鼠标右键，在弹出的菜单中选择 New Page 选项，就可以添加选项卡。根据程序需要多次单击 New Page 选项添加多个选项卡。如图 3-23 所示的 PageControl 中添加了 4 个选项卡。每个选项卡都是一个 TabSheet，如图 3-24 所示。各个选项卡之间互不影响，独立工作。

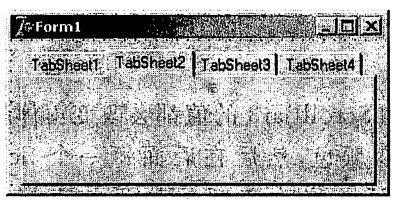


图 3-23 添加了 4 个选项卡的 PageControl

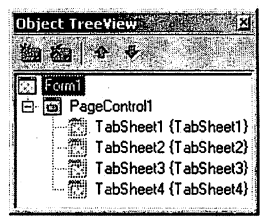


图 3-24 增加的 TabSheet

2. PageControl 组件的应用

【例 3-20】 使用选项卡来显示 3 首唐诗。

分析：可以使用 PageControl 来显示唐诗，并添加 3 个选项卡，每个选项卡就是一个容器，可以在不同的选项卡中添加不同的唐诗。

步骤如下：

(1) 添加 PageControl 组件 PageControl1 到窗体，并调整大小。选定 PageControl1，鼠标右键单击 PageControl1，选择 New Page 菜单项添加 3 个选项卡。

(2) 选择 TabSheet1，设置 Caption 为“静夜思”；选择 TabSheet2，设置 Caption 为“春晓”；选择 TabSheet3，设置 Caption 为“登鹳雀楼”。

(3) 选择 TabSheet1，在里面添加标签 Label1，调整 Label1 的大小。选择 TabSheet2，在里面添加标签 Label2，调整大小。选择 TabSheet3，在里面添加标签 Label3，调整大小。设置 Label1、Label2 和 Label3 的字体为隶书，大小为 16 号。

(4) 编写程序如下：

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 label1.Caption:='静夜思'+chr(13);
```

```
  label1.Caption:=label1.Caption+'床前明月光,'+chr(13);
  label1.Caption:=label1.Caption+'疑是地上霜;'+chr(13);
  label1.Caption:=label1.Caption+'举头望明月,'+chr(13);
  label1.Caption:=label1.Caption+'低头思故乡。';
  label2.Caption:='春晓'+chr(13);
  label2.Caption:=label2.Caption+'春眠不觉晓,'+chr(13);
  label2.Caption:=label2.Caption+'处处闻啼鸟;'+chr(13);
  label2.Caption:=label2.Caption+'夜来风雨声,'+chr(13);
  label2.Caption:=label2.Caption+'花落知多少。';
  label3.Caption:='登鹳雀楼'+chr(13);
  label3.Caption:=label3.Caption+'白日依山尽,'+chr(13);
  label3.Caption:=label3.Caption+'黄河入海流;'+chr(13);
  label3.Caption:=label3.Caption+'欲穷千里目,'+chr(13);
  label3.Caption:=label3.Caption+'更上一层楼。';
end;
```

程序运行结果如图 3-25 所示。

图 3-25　程序运行结果

3.9　小结

本章重点讲述了窗体以及 Delphi 中最常见的组件，主要有文本显示和编辑组件、分组的组件、按钮类组件、列表类组件、计时器、滚动条和多选项卡。同时，本章还详细介绍了上述基本组件的常用属性、方法和事件。读者应该学会使用这些组件，特别是要掌握其常用的属性、方法和事件。

习题

读者在编写下列程序的时候，除了要求程序正确之外，还应该尽量使得程序界面美观、操作方便、容错性强。

1. 在编辑框中输入长方体的长、宽、高，计算长方形的体积和表面积，程序界面如图 3-26 所示。

2. 用户登录程序。本程序给用户 3 次机会，如果在 3 次之内还不能正确输入用户名(不分大小写的“abc”)和口令(“123456”)，则程序提示非法用户，并停止运行，如果用户名和口令都对则提示“欢迎使用本系统”。另外，要求为两个编辑框设置加速键。界面如

图 3-27 所示。

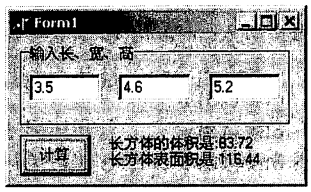


图 3-26 计算长方体表面积和体积的界面

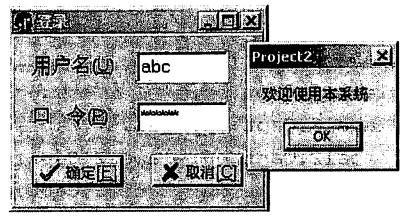


图 3-27 登录界面

提示：本程序可以使用 Tag 属性来记录用户名或者口令输入错误的次数，它的作用相当于一个全局变量。

3. 对第 2 题进行如下改进，要求输入口令后，无须按“确定”按钮直接按回车键也可以执行 BitBtn1(“确定”按钮)中的代码。另外，无须按“取消”按钮，只需要按 Esc 键也可以执行 BitBtn2 中的代码。请读者设置相应的属性来实现此功能。

4. 单位发工资，要求给出应发工资 x 元，程序能计算出钞票总张数最少的付款方案。程序运行界面如图 3-28 所示。

5. 在窗体上显示一个数字时钟，能够按照 12 小时制和 24 小时制显示当前的时间，界面如图 3-29 所示。

图 3-28 工资发放方案

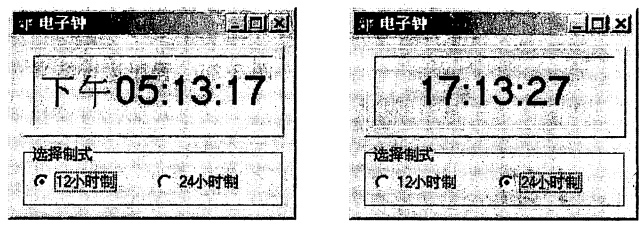


图 3-29 电子钟界面

提示：本处可以使用函数 formatdatetime(format,time)。Format 可以是'ampmhh:nn:ss'或者是'hh:nn:ss'两种格式。

6. 电话号码抽奖程序。输入若干个电话号码到组合框的下拉框，按“抽奖”按钮即可进行抽奖，将中奖号码显示在右边列表框，并从左边组合框中删除该号码。程序运行界面如图 3-30 所示。图的左边是一个 ComboBox，设置它的 Style 属性，使得它的外观样式如

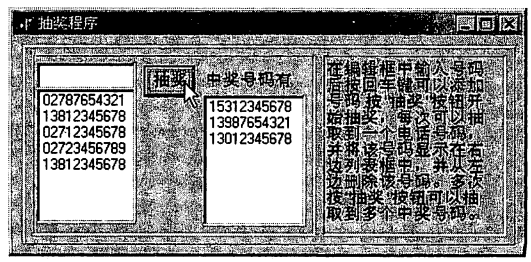


图 3-30 三次抽奖后的程序界面

图 3-30 所示。

7. 改进加法练习器程序。要求程序给出百分制的成绩；要求做错的题目不能再继续重做，而是更正后显示在列表框中。界面如图 3-31 所示。

8. 输入职工的信息，包括姓名、性别、职称和爱好。其中，姓名以编辑框的形式输入；性别以单选按钮的形式输入；职称包括教授、副教授、讲师、助教和见习，职称以组合框的形式输入；爱好是可选项，以复选按钮的形式给出。将输入的信息显示在 Memo 组件中。界面如图 3-32 所示。

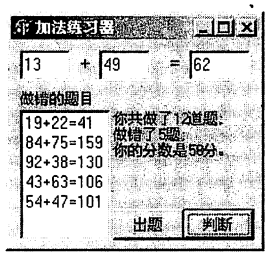


图 3-31　加法练习器

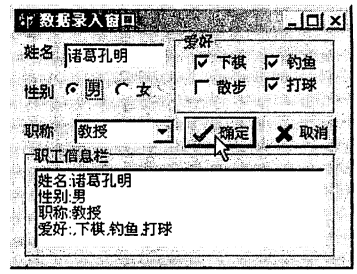
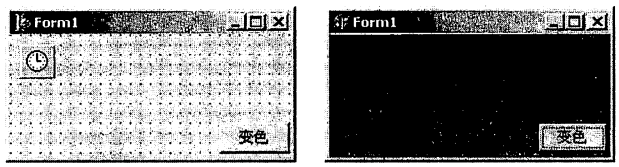


图 3-32　职工信息数据录入窗口

9. 编写一个程序，随机改变窗体的颜色。界面如图 3-33 所示。



(a)　(b)

图 3-33　改变窗体颜色的程序界面((b)是运行后的界面)

提示：可以使用 Rgb(r,g,b)函数来控制窗体的颜色，目前还没有接触到全局变量，因此可以考虑 Tag 属性。语句“form1. Color := rgb(form1. Tag, timer1. Tag, button1. Tag);”可以用来控制窗体的颜色，随时改变 3 个 Tag 属性值，就可以改变颜色。另外 r, g, b 都不会超过 255，而类似“a :=(a+3) mod (n+1)”语句可以使 a 无论如何变化都不会超过 n。

10. 设计一个界面实现类似界面折叠与展开的功能。在窗体上添加 Panel、Memo、Splitter、SpeedButton 等组件。单击按钮可以完成 Memo 的折叠和展开功能。界面如图 3-34 所示。

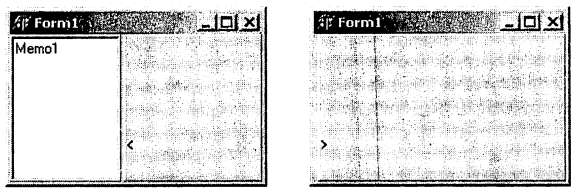


图 3-34　界面的折叠与展开

11. 编写程序实现自右至左反复滚动字幕。要求可以通过滚动条控制字幕滚动速度。界面如图 3-35 所示。

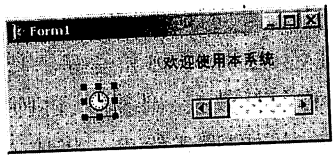


图 3-35 滚动字幕界面

第 4 章

程序控制结构

在程序设计中，常常采用三种流程控制结构，分别是顺序结构、分支结构（又叫选择结构）和循环结构。流程控制结构是结构化程序设计的基本思想。一个复杂的问题一般可以转化为有限个顺序、分支或循环结构来解决。

顺序结构是使用最为广泛的控制结构，程序从上往下顺序执行，即使是分支结构和循环结构，其内部也往往采用顺序结构。另外，各个模块之间也是从上向下顺序执行的。

分支结构是指程序执行到某个地方之后有多个分支，程序需要根据某个条件来决定到底执行哪个分支的一种流程控制结构。正是因为有了分支结构，程序和用户之间的交互能力才得以大大增强。在 Delphi 中提供了两种分支结构的语句，分别是 If 语句和 Case 语句。

循环结构通常使用在程序中需要重复执行某些语句和程序段的地方，它可以避免重复书写相同或类似的程序代码，使程序更加清晰明了和简化，提高了效率。在 Delphi 中提供了三种循环语句，分别是 While 语句、Repeat 语句和 For 语句。

4.1 分支结构

分支结构通过判断条件表达式，再根据结果进行不同的操作。条件表达式一般是逻辑表达式或关系表达式，类型为 Boolean 型，其值为 True 或 False。Boolean 类型是顺序型，False 的序号为 0，True 的序号为 1。

4.1.1 If 语句

If 语句是最常见的分支结构语句，其操作是如果所给条件成立（值为 True）则执行＜语句序列 1＞，否则执行＜语句序列 2＞，其流程图如图 4-1 所示。

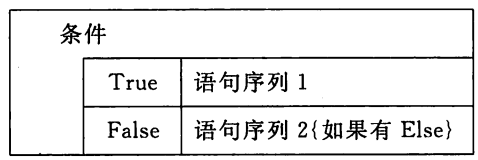


图 4-1 If 语句的流程图

1. If 语句的语法格式

If 语句的语法格式为：

```
If <条件>
Then <语句序列 1>
[Else <语句序列 2>]
```

说明：

(1) 条件是关系表达式或逻辑表达式，程序是这样判断的，如果条件成立(值为 True)，则执行<语句序列 1>，如果条件不成立(值为 False)则执行<语句序列 2>。

(2) <语句序列 1>和<语句序列 2>都可以省略。

(3) 如果<语句序列 1>或<语句序列 2>都是几个语句，则用 Begin 和 End 括起来形成复合语句的形式。

(4) 将整个 If 语句看成一个语句，因此不能在 If 语句中间加分号。

例如：

```
If a>b
Then writeln(a);          //这个分号有错误
Else writeln(b);
```

再看下面：

```
If a>b
Then begin
    t:=a;
    a:=b;
    b:= t;                //可以加分号，和 Begin、End 形成复合语句
    end                   //不可加分号，整个 If...Then...Else 是一个语句
else begin
    writeln('1111');
    writeln('2222');
    end;
```

2. If 语句的应用

【例 4-1】 求 3 个数的最大值。

分析：用 If 语句来编写，可以使程序比较简单。程序如下：

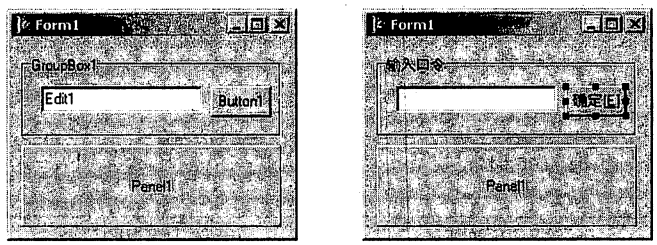
```
program Project1;

{$APPTYPE CONSOLE}

uses
  SysUtils;
 var ma,a,b,c:real;

begin
 write('a,b,c=?');
```

```
 readln(a,b,c);
 ma:=a;                    //假设 a 最大
 if b>ma then ma:=b;       //若 b>ma,则最大为 b
 if c>ma then ma:=c;       //若 c>ma,则最大为 c
 writeln('最大数为',ma);
 readln;
end.
```

【例 4-2】 简单的密码程序。

程序设计步骤：

(1) 添加组件 GroupBox、Edit、Button 和 Panel 到窗体，设置组件的属性：GroupBox1 的 Caption 设置为“输入口令”，Edit1 的 Passwordchar 设置为“*”，设置 Button1 的 Caption 为“确定[&E]”，界面如图 4-2(a)、(b)所示。



(a) 设置属性前的程序界面　　(b) 设置属性后的程序界面

图 4-2　设置属性前后的程序界面

(2) 编写 Button1 的 OnClick 事件过程如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 if uppercase(edit1.Text)='ABCDEF'          //口令是“abcdef”,不分大小写
 then panel1.Caption:='欢迎使用本系统'
 else begin
     edit1.Text:='';
     edit1.SetFocus;                        //清空文本框并置焦点
     panel1.Caption:='口令错误';
     end;
end;
```

另外，我们希望输入口令后按回车键也可以判断口令是否正确，因此编写 Edit1 的 OnKeyPress 事件过程如下：

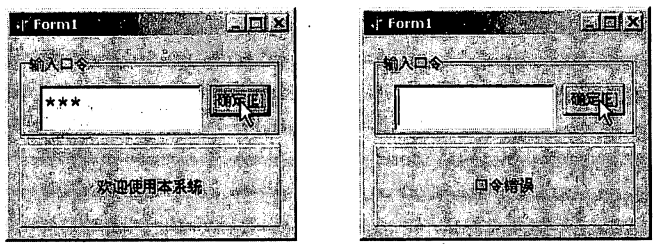
```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
 if key=#13 then                            //如果按了回车键
 begin
  if uppercase(edit1.Text)='ABCDEF'
  then panel1.Caption:='欢迎使用本系统'
  else begin
       edit1.Text:='';
```

```
        edit1.SetFocus;
        panel1.Caption:='口令错误';
      end;
 end;
end;
```

下面的 Edit1 的 OnChange 事件过程用于清除 Panel1 中的内容：

```
procedure TForm1.Edit1Change(Sender: TObject);
begin
 panel1.Caption:='';
end;
```

(3) 按 F9 键运行程序，界面如图 4-3(a)、(b)所示。



(a) 输入正确的口令　　(b) 输入错误的口令

图 4-3　输入口令界面

3. If 语句的嵌套

如果在 If 语句中还有 If 语句就称之为 If 语句的嵌套。有时 If 语句的嵌套会产生歧义。例如：

```
If yuwen>=60
Then if yuwen>=85
    Then writeln('优秀')
Else writeln('不及格');
```

我们本希望 Else 子句属于前面一个 If 语句，但是实际上本例中的 Else 子句属于后面一个 If 语句。Delphi 规定：Else 子句总是属于与它最靠近的那个没有 Else 子句的 If...Then 语句。虽然如此，用户还是比较容易理解错误。因此，建议把上面的这个嵌套 If 语句写成如下形式。

```
If yuwen>=60
Then if yuwen>=85
    Then writeln('优秀')
    Else
Else writeln('不及格');
```

或者：

```
If yuwen>=60
Then Begin if yuwen>=85
```

```
        Then writeln('优秀')
    End
Else writeln('不及格');
```

这样就不容易理解错误了。另外，建议书写程序时将程序行适当缩进，这样也可以避免对程序的错误理解。

【例 4-3】 修改上面的密码程序，允许用户 3 次出错，如果第 3 次密码仍然错误，则结束程序。请使用嵌套的 If 语句完成。

只需要简单 Button1 的 OnClick 事件过程即可。

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 if uppercase(edit1.Text)='ABCDEF'
 then panel1.Caption:='欢迎使用本系统'
 else begin
      edit1.Tag:=edit1.Tag+1;          //无须重新定义变量
      if edit1.Tag=3 then
       begin
        messagedlg('3 次密码错误,您无权使用本程序',mterror,[mbok],1);
        //显示错误信息
        application.Terminate;         //程序停止
       end;
       edit1.Text:='';                 //此处无须加 Else
       edit1.SetFocus;
       panel1.Caption:='口令错误';
      end;
end;
```

不要忘记将 Edit1 的 OnKeyPress 事件过程也做类似修改。此处省略。

4.1.2 Case 语句

If 语句可以解决两路分支问题，而 Case 语句可以解决多路分支问题，即从多个语句序列中选择一路语句序列执行。

1. Case 语句的语法格式

Case 语句的语法格式为：

```
Case <表达式>of
[常量表 1: [语句序列 1;]]
[常量表 2: [语句序列 2;]]
[常量表 3: [语句序列 3;]]
    ⋮
[常量表 n: [语句序列 n;]]
[Else [语句序列 n+1;]]
End;
```

说明：

(1) 表达式必须是顺序类型，如整型、字符型、枚举、子界、布尔型等。

(2) 各个常量表中的常量应该用逗号隔开,且不能相同。

(3) 如果语句序列是多个语句,可以使用 begin 和 end 括起来形成复合语句。

(4) 如果表达式的值与某个常量表中的常量值相等则程序就执行该语句序列,执行完毕,程序转到 Case 语句之外执行其他余句。如果表达式的值与所有常量表中的常量值都不相等,则程序执行 Else 之后的语句序列 n+1(如果有 Else 部分)。

Case 语句的流程图如图 4-4 所示。

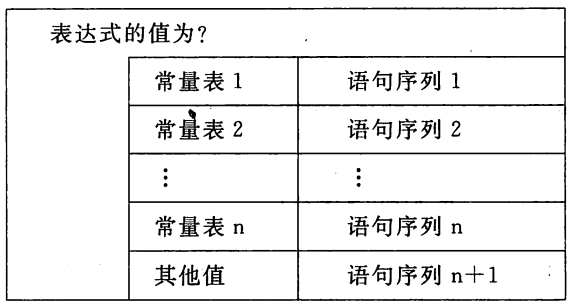


图 4-4 Case 语句的流程图

2. Case 语句的应用

【例 4-4】 计算每个月的天数。

分析:根据常识 1 月、3 月、5 月、7 月、8 月、10 月和 12 月都有 31 天;而 4 月、6 月、9 月和 11 月都有30 天;2 月闰年有 29 天,平年有 28 天。符合下列情况之一就是闰年:①年号能被 400 整除;②年号能被 4 整除但是不能够被 100 整除。

建立 Console Application 应用程序如下:

```
program Cadays;

{$APPTYPE CONSOLE}
uses
 SysUtils;
 var year,month,days:integer;

begin
 write('year,month=?');
 readln(year,month);
 case month of
  1,3,5,7,8,10,12:days:=31;
  4,6,9,11:days:=30;
  2:if (year mod 400=0)or(year mod 100<>0)and(year mod 4=0)
    then days:=29
    else days:=28;
  else writeln('Input year error!');
 end;
 writeln(year:4,'年',month:1,'月有',days:2,'天。');
 readln;
end.
```

4.2　循环结构

Delphi 提供了三种循环结构的语句，分别是 While 语句、Repeat 语句和 For 语句。这三种语句分别在不同的环境条件下使用，但一般情况下，它们可以互相替代。三种循环语句各有特色，其中 While 语句和 Repeat 语句可用于循环次数未知的情况下，根据循环条件来控制循环次数，而 For 语句可用于循环次数已知的情况。在实际应用中，用户需选择合适的循环语句，对于某些问题，三种循环都可以实现。

4.2.1　While 语句

While 语句属于前测型循环语句。首先判断条件，根据条件来决定是否循环，如果条件成立则循环，否则不循环，也有可能一次也不循环。其流程图如图 4-5(a)、(b)所示。

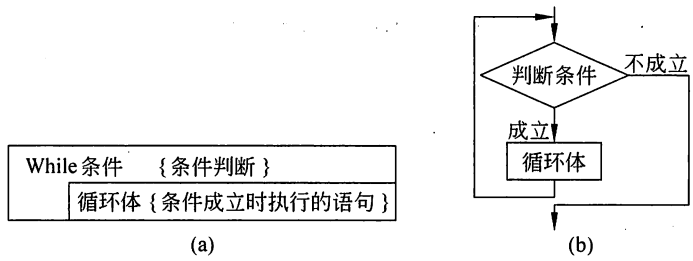


图 4-5　While 语句的流程图

1. While 语句的语法格式

While 语句的语法格式为：

```
While <条件> Do
  [循环体];
```

说明：

(1) 如果条件成立(值为 True)，则执行循环体；如果条件不成立(值为 False)，则结束循环，执行循环体之后的语句。

(2) 循环体可以是若干个语句，如果是一个以上的语句，可以使用 Begin 和 End 括起来，形成复合语句的形式。

(3) 可以在循环体中添加 Continue 语句，用 If 语句来控制 Continue。程序执行 Continue 用于结束本次循环，执行下一次循环。

(4) 还可以在循环体中添加 Break 语句，用 If 语句来控制 Break。程序执行 Break 用于结束整个循环，执行循环体之后的语句，程序结束循环之前无须判断循环条件。

2. While 循环的应用

【例 4-5】 求 s=1+2+3+…+100 的值。

分析：可以使用变量 n 来作为加数，其变化从 1 一直计数到 100，每一次将 n 的值加到 s 中，n 都要自加 1。s 用来保存 1+2+3+…+100 的和，s 的初始值赋为 0。流程图如

图 4-6 所示。

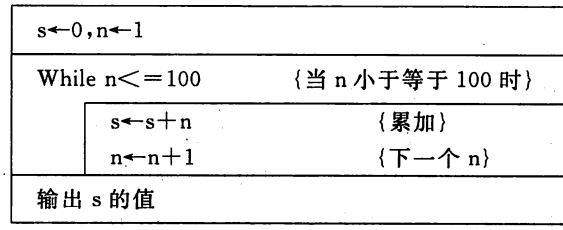


图 4-6 求 s 值的流程图

程序设计步骤：

(1) 添加组件到窗体，设置组件的属性，界面如图 4-7 所示。

(2) 编写 Button1 的 OnClick 时间过程。

```
procedure TForm1.Button1Click(Sender: TObject);

 var s,n:integer;

begin
 s:=0;n:=1;
 while n<=100 do
 begin
  s:=s+n;
  n:=n+1;
 end;
 panel1.Caption:='s=1+2+3+…+100='+inttostr(s);
end;
```

(3) 按 F9 键运行程序，界面如图 4-8 所示。

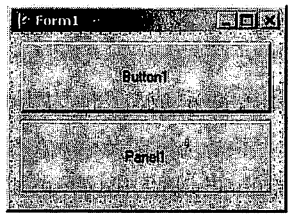


图 4-7 程序界面

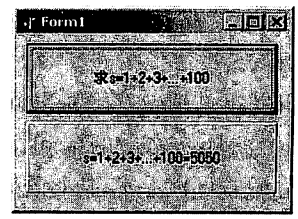


图 4-8 程序运行界面

4.2.2 Repeat 语句

Repeat 语句一般用于循环次数不确定的循环。首先执行循环语句，然后通过判断循环条件来决定是否继续执行循环，因此 Repeat 语句至少要执行一次循环。流程图如图 4-9(a)、(b)所示。

1. Repeat 语句的语法格式

Repeat 语句的语法格式为：

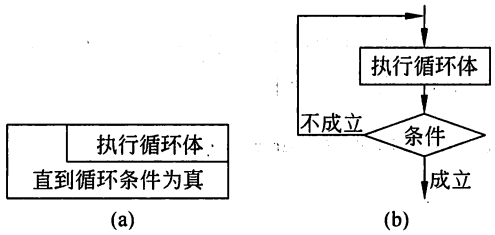


图 4-9　Repeat 语句的流程图

```
Repeat
  循环体;
Until <条件>;
```

说明：

(1) Repeat 循环是后侧型循环，首先执行循环体，再判断循环条件。

(2) 语句体可以是若干个语句，多条语句无须用 Begin 和 End 括起来，可用自身的 Repeat 和 Until 将其括起来。

(3) 循环条件的判断是：如果条件为真则结束循环，转而执行循环体之后语句；如果循环条件为假则继续循环。一般循环条件在执行的过程中会改变。

(4) 可以在循环体中加入 Continue 语句，一般用 If 语句来控制 Continue。Continue 语句可以让程序提前结束本次循环(无须判断循环条件)，转而去执行下一次循环。

(5) 还可以在循环体中加入 Break 语句，一般使用 If 语句来控制 Break。Break 语句可以让程序结束整个循环(无须判断循环条件)，转而去执行循环体之后的语句。

2. Repeat 循环的应用

【例 4-6】 求 π 的值。利用公式 π/4≈1－1/3＋1/5－1/7＋…求近似值，直到最后一项的绝对值小于 0.0001 为止。

分析：先计算出右边多项的和，然后再乘以 4，即为所求 π 的近似值。令右边各项之和的值为 pi，右边每一项的分母分别为奇数 1、3、5…，符号是一正一负。可以使用 n 作为计数器，控制分母，fh 用来控制正负，则每一项的值为 fh/n，使用循环将每一项加到 pi 中就可以得到右边各项之和。最后乘以 4，即可得到 π 的近似值。根据分析绘制出流程图，如图 4-10 所示。

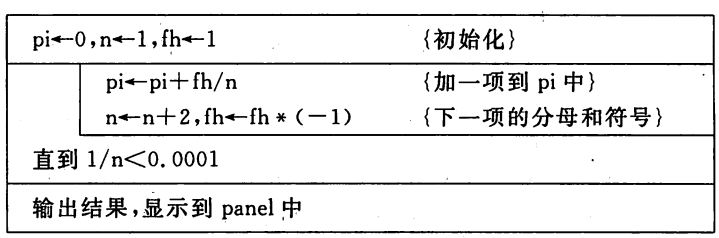


图 4-10　求 π 近似值的流程图

程序设计步骤：

(1) 添加组件到窗体，设置组件的属性，界面如图 4-11 所示。

(2) 编写 Button1 的 OnClick 事件过程如下：

```
procedure TForm1.Button1Click(Sender: TObject);

 var fh,pi,n:real;

begin
 pi:=0; n:=1;fh:=1.0;
 repeat
  pi:=pi+fh/n;
  n:=n+2;          //分母为奇数
  fh:=-fh;         //每项的符号是变化的
 until 1/n<=0.0001;
 pi:=4*pi;
 panel1.Caption:='π='+floattostr(pi)
end;
```

(3) 按 F9 键运行程序，界面如图 4-12 所示。

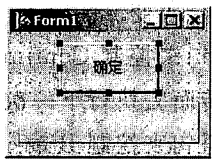


图 4-11 程序界面

图 4-12 程序运行结果

【例 4-7】 求 π 的值。利用公式 $\frac{\pi}{2}=\frac{2}{\sqrt{2}}\cdot\frac{2}{\sqrt{2+\sqrt{2}}}\cdot\frac{2}{\sqrt{2+\sqrt{2+\sqrt{2}}}}\cdot\cdots$，当相邻的两个 π 值小于文本框中给定的数值时，结束运算，输出此时的 π 值。

分析：设 $f(1)=\frac{2}{\sqrt{2}}$，$f(2)=\frac{2}{\sqrt{2}}\cdot\frac{2}{\sqrt{2+\sqrt{2}}}$，第 1 项的分母 $p_1=\sqrt{2}$，第 2 项的分母 $p_2=\sqrt{2+\sqrt{2}}$，…，第 n+1 项的分母 $p_{n+1}=\sqrt{2+p_n}$，因此，根据上述分析有 $f(n+1)=f(n)*\frac{2}{p_{n+1}}$，最后所求 π 的值为 2 * f(n)。根据分析，画出解题的流程图，如图 4-13 所示。

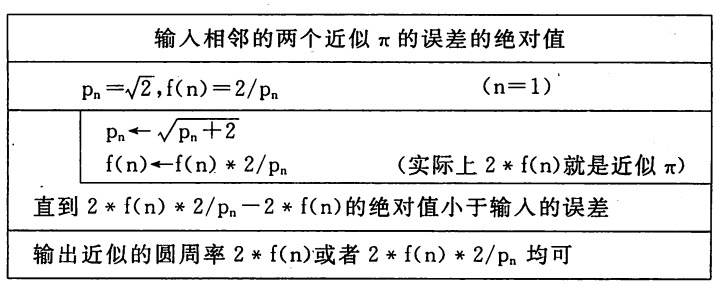


图 4-13 求 π 近似值的流程图

程序设计步骤：

(1) 添加组件设置属性，设计界面如图 4-14 所示。

(2) 根据上面的分析和流程图写出 Edit1 的 KeyPress 过程，程序如下：

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);

 var pn,fn:real;

begin
 if key=#13 then                    //输入误差后按回车键
 begin
  pn:= sqrt(2); fn:=2/pn;           //第 1 项的分母和第 1 项
  repeat
   pn:=sqrt(pn+2);                  //第 n 项的分母
   fn:=fn * 2/pn;                   //n 项乘积，再乘以 2 就是 π 的近似值
  until abs(2 * (fn * 2/pn)-2 * fn) <=strtofloat(edit1.Text);
                                    //相邻的两个近似 π 分别为 2 * (fn * 2/pn)和 2 * fn
  panel1.Caption:='π='+floattostr(2 * fn);
 end;
end;
```

(3) 按 F9 键运行程序，界面如图 4-15 所示。

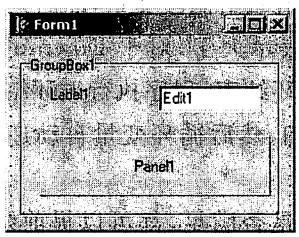


图 4-14　程序界面

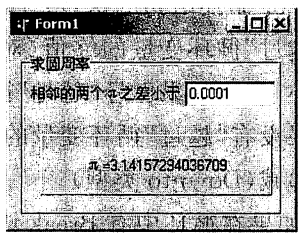


图 4-15　程序运行结果

4.2.3　For 语句

在循环次数不确定的时候，使用 While 循环或 Repeat 循环比较好；如果循环次数确定，最好使用 For 循环。与前两个循环不同的是，For 循环中有一个循环变量用于控制循环次数，每循环一次该变量就会自动增加或者减小。

For 语句的流程图如图 4-16(a)、(b)所示。

1. For 语句的格式

For 语句的格式为：

```
For <循环变量>:=<初值>{To|Downto} <终值>Do
[循环体];
```

说明：

(1) <循环变量>为必要参数，它必须是顺序类型，用作循环次数的计数器，并且在循环体中不允许人为地改变循环变量的值。

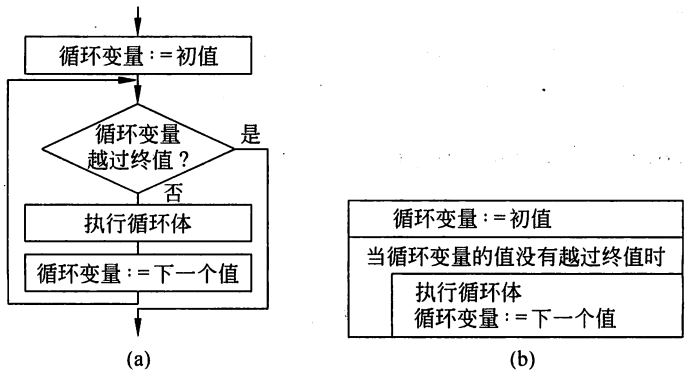


图 4-16 For 语句的流程图

(2) <初值>和<终值>是循环变量的初始取值和终值,它可以是变量或表达式,但是类型必须和循环变量保持一致。如果循环变量是表达式,则在进入循环的时候,表达式的值应该是确定的。

(3) To 表示循环计数器递增,Downto 表示循环变量计数器递减。

(4) 循环体可以是一个语句或者多个语句,如果是多个语句可以使用 Begin 和 End。

(5) For 循环语句的执行:首先将循环变量的初始值赋给循环变量,然后判断循环变量是否"越过"循环变量的终值(对于 To 类型,"越过"表示大于;对于 Downto 类型,"越过"表示小于)。如果已经"越过"循环变量的终值,则循环结束,程序执行循环结构之后的语句;如果没有"越过"循环变量的终值,则执行循环语句,然后循环变量自动递增(To 类型)或者递减(Downto 类型)。

(6) 如果在循环中有 Break 语句,则不论循环条件是否满足,都可以提前结束循环。可以使用 If 语句来控制 Break 语句。

(7) 可以在循环中加入 Continue 语句,Continue 语句的作用是结束本次循环,进入下次循环。可以使用 If 来控制 Continue 语句。

(8) Break 语句和 Continue 语句可以使 For 循环结构更加巧妙。

2. For 循环语句的应用

【例 4-8】 求 s=1+2+4+5+7+8+10+11+…,当 s 的值刚好大于 1000 时的值。

分析:本程序是求 s=1+2+4+5+7+8+10+11+…,当 s 的值刚好大于 1000 时的 s 的值,由于不知道循环多少次时 s 的值刚好大于 1000,因此可以把循环次数设计为 1000 次,然后在循环的过程中根据条件中途退出循环,中途退出循环的条件是 s>1000。另外,本程序求和的时候没有包括 3 的倍数,所有 3 的倍数都排除在外,因此,在加的过程中如果判断加数是 3 的倍数则结束本次循环,进入下一次循环。流程图如图 4-17 所示。

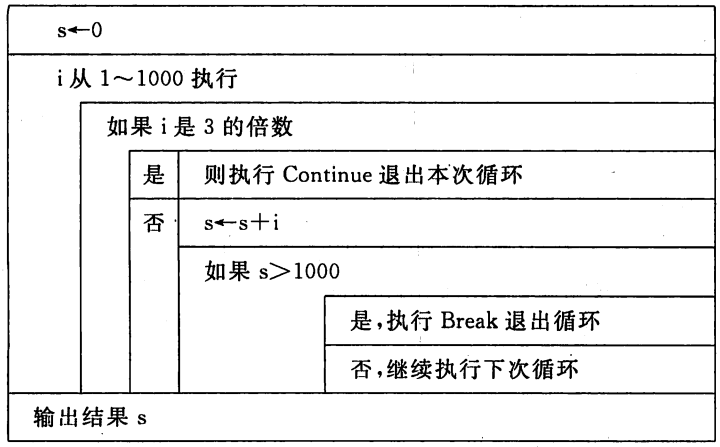


图 4-17　求 s 值的流程图

程序设计步骤如下：

(1) 界面设置，属性设置。界面如图 4-18 所示。

(2) 编写 Button1 的 OnClick 事件过程。

```
procedure TForm1.Button1Click(Sender: TObject);

 var s,i:integer;

begin
 s:=0;
 for i:=1 to 1000 do
 begin
  if i mod 3=0 then continue;
  s:=s+i;
  if s>1000 then break;
 end;
 panel1.Caption:='1+2+4+5+…+';
 panel1.Caption:=panel1.Caption+inttostr(i)+'='+inttostr(s);
end;
```

(3) 按 F9 键运行程序，界面如图 4-19 所示。

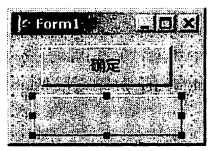


图 4-18　程序界面

图 4-19　程序运行结果

4.2.4　多重循环

如果在循环中还有循环，就构成多重循环结构。常用的多重循环有二重循环和三重循环。可以使用相同的循环语句或不同的循环语句来实现多重循环。

【例 4-9】 求 100 以内的素数。

分析：判断 n 是否为素数可以这样进行，先假设 n 是素数，设一个标志 flag 的值为 True。然后在 2 到$\sqrt{n}$之间寻找 n 的因子，只要能够找到 n 的因子(一个足够)，就令 flag 的值为 False。如果经过 2 到$\sqrt{n}$的循环 flag 的值仍然是 True，说明 n 是素数，否则 n 不是素数。如果 n 是素数，则输出这个 n。

由于 1 不是素数，让 n 从 2 到 100 循环，执行上面的操作，即可输出 100 以内所有的素数。本程序的循环次数已知，因此两个循环都用 For 语句来实现。

根据以上分析，不难画出本程序的流程图，如图 4-20 所示。

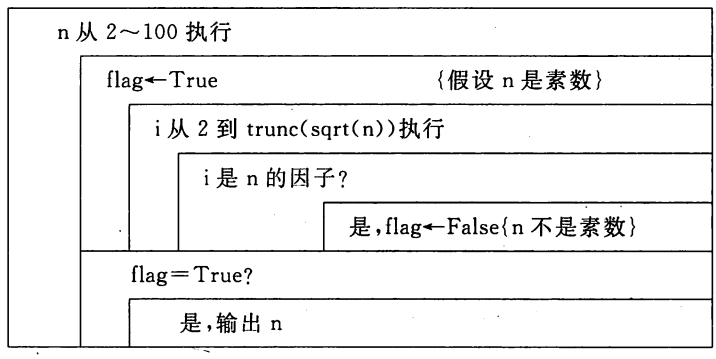


图 4-20 求 100 以内所有素数的流程图

程序设计步骤如下：

(1) 添加组件，属性设置省略，界面如图 4-21 所示。

(2) 编写 Button1 的 OnClick 事件过程。

```
procedure TForm1.Button1Click(Sender: TObject);

 var n,i:integer;
     flag:boolean;

begin
 for n:=2 to 100 do
 begin
  flag:=true;                          //假设 n 是素数
  for i:=2 to trunc(sqrt(n)) do
                                       //在 2 到 trunc(sqrt(n))之间寻找 n 的因子
                                       //循环变量的初值和终值为整型,因此要取整
  if n mod i=0 then flag:=false;
                                  //i 是 n 的因子,因此 n 不是素数,故修改 flag 为 false
  if flag then
  listbox1.Items.Add(inttostr(n)); //输出素数 n
 end;
end;
```

(3) 按 F9 键运行程序，界面如图 4-22 所示。

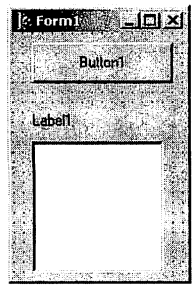


图 4-21　程序界面

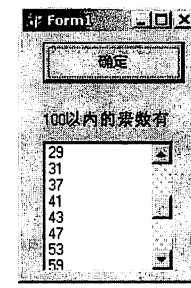


图 4-22　程序运行结果

【例 4-10】 验证哥德巴赫猜想，哥德巴赫认为一个非常大的偶数可以分解成为两个素数之和的形式。本程序要求输入一个较大的偶数 n，将这个偶数 n 分解成为两个素数 p 和 q 相加的形式，例如，输入 10，输出 10＝3＋7 和 10＝5＋5。试编写程序。

分析：一个数 n 可以分解成 p＋q 的形式：p 从 2 到 n/2 循环，而 q 始终取值为 n－p，只要满足 p 和 q 都是素数就输出 p 和 q。判断 p 和 q 是否是素数，可以使用 For 循环，外加 p 从 2 到 n/2 的循环，这样使用二重循环即可解答本题。根据分析画出流程图，如图 4-23 所示。

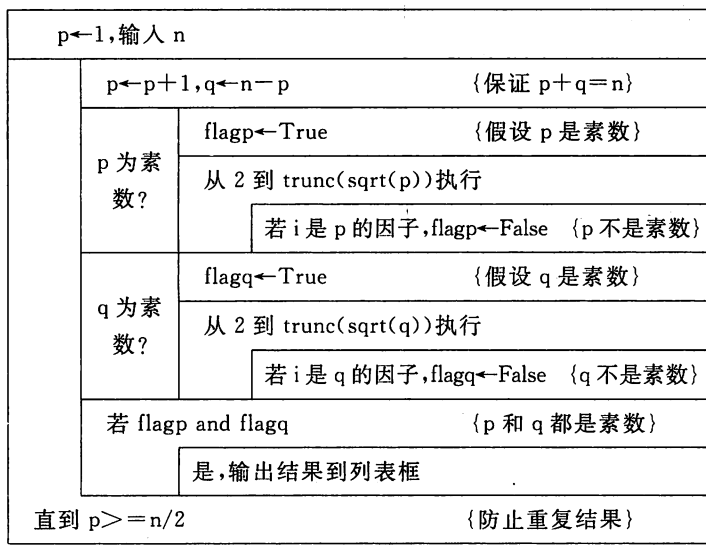


图 4-23　验证哥德巴赫猜想的流程图

程序设计步骤如下：

(1) 添加组件，属性设置省略，界面如图 4-24 所示。

(2) 编写 Button1 的 OnClick 事件过程。

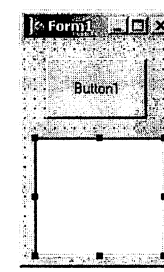


图 4-24　程序界面

```
procedure TForm1.Button1Click(Sender: TObject);

 var n,p,q,i:integer;
     flagp,flagq:boolean;
     s:string;
begin
```

```
 listbox1.Items.Clear;
n:=strtoint(inputbox('输入一个正整数 n','输入 n ',''));
 p:=1;
 repeat
  p:=p+1; q:=n-p;
  flagp:=true;                          //假设 p 是素数
  for i:=2 to trunc(sqrt(p))do
   if p mod i=0 then flagp:=false;      //p 不是素数
  flagq:=true;                          //假设 q 是素数
  for i:=2 to trunc(sqrt(q)) do
   if q mod i=0 then flagq:=false;      //q 不是素数
  if flagp and flagq then
   begin
    s:=inttostr(n)+'='+inttostr(p)+'+'+inttostr(q);
    listbox1.Items.Add(s);
   end;                                 //输出
  until p>=n/2;
 end;
```

(3) 按 F9 键运行程序，界面如图 4-25 所示。

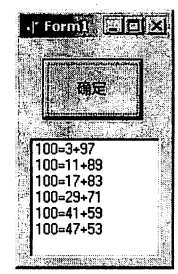


图 4-25 程序运行结果

在设计循环类程序时，一定要思路清晰，最好画出流程图，对于多重循环，应该注意以下几个问题：

(1) 内外循环只能嵌套，不能交叉。

(2) 不同层的循环，不要使用相同的循环变量，并列的循环除外。

(3) 注意程序的缩进格式，这样可以减少循环程序出错的概率。

4.3 小结

本章讲述了程序的控制结构，包括顺序结构、分支结构和循环结构。本章的各个控制结构都安排有典型的例题，通过这些例题，读者应该掌握这三种基本控制结构的用法，并能够在程序中灵活运用。

读者应注意 If 语句和 Case 语句的区别和适用场合；注意三种循环语句的特点和优点；在何种情况下适合使用哪种循环。这些读者都应该熟知，并灵活掌握和使用。多重循环是难点，应注意其中的注意事项。

习题

1. 有下面一段代码：

```
a:=6;b:=20;
If a>5
Then If a<4
```

```
    Then b:=10
    Else
Else b:=15;
```

程序执行后，b 的值为_____。

A. 20　　B. 10　　C. 15　　D. 其他值

2. 有下面程序段：

```
year:=2008;month:=2;
Case month of
 1,3,5,7,8,10,12:days:=31;
 4,6,9,11:days:=30;
 2: Begin days:=28;
       If year mod 400=0 then days:=29;
       If (year mod 100<>0) and (year mod 4=0) then days:=29
   End;
End;
```

程序执行后，days 的值为_____。

A. 28　　B. 29　　C. 30　　D. 31

3. 输入某个学生的成绩，输出该生成绩的等次，已知：85 分或者 85 分以上为“优秀”，70 分（含 70 分）到 85 分为“良好”，60 分（含 60 分）到 70 分为“及格”，60 分以下为“不及格”，试编写程序。

4. 输入某年、月、日，判断该年、月、日是这年的第几天。

5. 以 1、2、3、4、5 为边长，可以形成多少个三角形？假设三角形的三边为 a、b、c，a=2、b=2、c=3 和 a=2、b=3、c=2 以及 a=3、b=2、c=2 表示的是一个三角形。类似的情况，在统计的时候只能按照一个三角形来计算。

提示：不妨令 a≤b≤c，这样就不会出现把 a=2、b=2、c=3 和 a=2、b=3、c=2 以及 a=3、b=2、c=2 看成是三个三角形了。

6. 打印如图 4-26 所示的图形。

7. 编写程序，求出等差级数 a+2a+3a+…+na 的值，n 和 a 是由键盘输入的正整数。尝试用多种方法完成。

8. 编写程序，判断一个数是否为完数，所谓完数，就是这个数刚好等于它的因子之和，如 6 的因子为 1、2、3，而 6=1+2+3，所以 6 为完数。

9. 输入两个正整数 m、n，求 m 和 n 的最小公倍数和最大公约数。这个问题在本章中已经讲解，读者可以考虑使用多种方法。

提示：求 m 和 n 的最大公约数相当于求 n 和 m mod n 的最大公约数，而任何正整数和 0 的最大公约数就是这个正整数。

10. 打印如图 4-27 所示的图形。

11. 求 $s=1+\frac{1}{4}+\frac{1}{9}+\frac{1}{16}+\frac{1}{25}+\cdots+\frac{1}{k^2}$，其中 k 从键盘输入。

图 4-26 第 6 题的打印图形

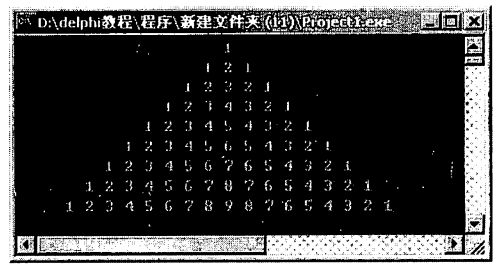


图 4-27 第 10 题的打印图形

12. 求 $f(x)=1-\frac{x^2}{3!}+\frac{x^4}{5!}-\frac{x^6}{7!}+\frac{x^8}{9!}-\cdots-\frac{x^{98}}{98!}$ 的值，其中 x 由键盘输入。

13. 著名的百鸡百钱问题，已知公鸡 5 元钱 1 只，母鸡 3 元钱 1 只，小鸡 1 元钱 5 只，问：100 元钱买 100 只鸡，公鸡、母鸡、小鸡各有多少只？

14. 两个乒乓球队进行比赛，各出 3 人。甲队为 A、B、C 三人，乙队为 X、Y、Z 三人。已抽签决定比赛对阵名单。有人向队员打听比赛名单，A 说他不和 X 比赛，C 说他不和 X、Z 比赛，试编写程序找出三队比赛选手。

15. 猴子吃桃问题。小猴子第一天摘下若干个桃子，当即吃掉一半，还不过瘾，又多吃了一个。第二天又将剩下的桃子吃了一半，并多吃了一个。以后每天都吃剩下的一半，并多吃一个。等到第 10 天早上，猴子想吃桃子时发现只剩下一个桃子。问第一天小猴子共摘了多少桃子？

第 5 章

枚举、子界与集合

Delphi 的数据类型分为标准类型和自定义类型，标准类型如整型、实型、字符型、布尔型。标准类型无须定义，可以直接用来定义变量。自定义类型分为枚举、子界、集合、数组、记录和文件。自定义类型必须先定义类型，后定义变量。本章将介绍枚举、子界和集合。

5.1 枚举类型

枚举类型就是将有限的数据用标识符的形式表现出来。例如，表示彩旗的颜色、教师的职称、每周的星期等。这些数据的可能值都有限，并且表示不方便、不直观。幸好 Delphi 提供的枚举类型可以很好地解决这个问题。

5.1.1 枚举类型及变量的定义

1. 枚举类型的定义

枚举类型定义的语法格式如下：

```
type
  <类型名称>=(<标识符 1>,<标识符 2>…<标识符 n>);
```

说明：

(1) ＜类型名称＞，就是用户自定义的枚举类型的名称。

(2) ＜标识符＞就是该类型的常量元素，它的个数有限，枚举类型是顺序类型。

(3) 枚举常量不允许重复出现在一个枚举类型中，也不允许同一个枚举常量出现在不同类型中。

例如：下面的代码就定义了一个具有 7 个常量的枚举类型。

```
Type
  Weekday=(Sun,Mon,Tue,Wed,Thu,Fri,Sat)
```

2. 枚举类型变量的定义

枚举类型变量的定义必须遵循先定义类型再定义变量的原则。例如：

```
var
  w1,w2,w3:weekday;
  a,b,c:integer;
```

Delphi 还允许把枚举类型和变量定义放在一起，但不建议这样做。如：

```
var
  w1,w2,w3:(sun,mon,tue,wed,thu,fri,sat);
  a,b,c:integer;
```

5.1.2 枚举类型的运算

枚举类型是顺序类型，可以使用顺序类型的函数来操作枚举类型，还可以使用关系运算来比较枚举值的大小。

(1) 用 pred 函数求枚举值的前导。例如，pred(mon)的值为 sun；pred(sat)的值为 fri；sun 没有前导等。

(2) 用 succ 函数求枚举值的后继。例如，succ(sun)的值为 mon；succ(fri)的值为 sat；sat 没有后继等。

(3) 用 ord 函数求枚举值的序号。例如，sun、mon、tue、wed、thu、fri、sat 的序号分别为 0～7；ord(mon)的值为 1 等。

(4) 用 low 函数可以求得第一个枚举值。low(weekday)的值为 sun(参数为类型)；low(w1)的值为 sun(参数是变量)；low(tue)的值为 sun(参数是常量)等。

(5) 用 high 函数可以求得最后一个枚举值，用法类似函数 low。

(6) 使用关系运算比较枚举值的大小。例如，sun＞mon 的值为 false；tue＞＝sun 的值为 true 等。

【例 5-1】 输入今天星期几(序号)，输出昨天星期几和明天星期几。

分析：关于星期问题，很容易想到枚举类型，因为星期几是有限的几个值。

设计步骤如下：

(1) 添加组件，设计界面，如图 5-1 所示。其中大的面板为 Panel1，小的面板从上到下分别是 Panel2 和 Panel3。编辑框为 edit1。

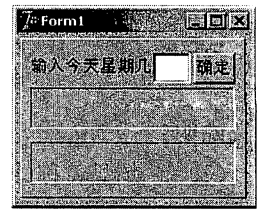


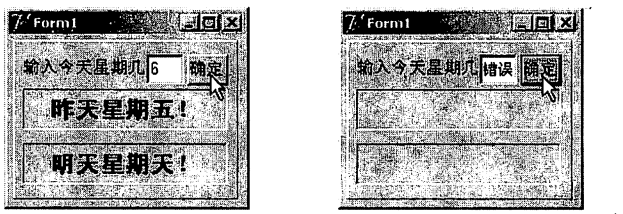
图 5-1 程序界面

(2) 编写程序代码如下：

```
type
 weekday=(sun,mon,tue,wed,thu,fri,sat);
procedure TForm1.Button1Click(Sender: TObject);

 label en;

 var
  w1:weekday;n:integer;

begin
 n:=strtoint(edit1.Text);
 if(n>6)or(n<0)
```

```
then begin
       edit1.Text:='错误';                          //输入的星期几必须是 0~6 之间
       panel2.Caption:='';
       panel3.Caption:='';
       goto en                                      //输入错误,结束程序,转到程序结尾
     end;
case(n-1+7)mod 7 of                                 //昨天的序号
 ord(sun):panel2.Caption:='昨天星期天!';
 ord(mon):panel2.Caption:='昨天星期一!';
 ord(tue):panel2.Caption:='昨天星期二!';
 ord(wed):panel2.Caption:='昨天星期三!';
 ord(thu):panel2.Caption:='昨天星期四!';
 ord(fri):panel2.Caption:='昨天星期五!';
 ord(sat):panel2.Caption:='昨天星期六!';
end;
case(n+1) mod 7 of                                  //明天的序号
 ord(sun):panel3.Caption:='明天星期天!';
 ord(mon):panel3.Caption:='明天星期一!';
 ord(tue):panel3.Caption:='明天星期二!';
 ord(wed):panel3.Caption:='明天星期三!';
 ord(thu):panel3.Caption:='明天星期四!';
 ord(fri):panel3.Caption:='明天星期五!';
 ord(sat):panel3.Caption:='明天星期六!';
end;
 en:
end;
```

(3) 程序运行结果如图 5-2 所示。



(a) 正确结果　　(b) 输入了 0~6 之外数字后的结果

图 5-2　程序运行结果

5.2　子界类型

在现实生活中很多数据的取值是有一定范围的，例如，每个月的天数在 28～31 天之间，月份在 1～12 之间，大写英文字母在 A～Z 之间等。像上述类型的数据可以将其定义为子界类型。定义为子界类型可以在一定范围内防止出错。子界类型也是顺序类型。

1. 子界类型的定义

子界类型的定义格式如下：

```
type
  <类型名称>=<常量 1>..<常量 2>;
```

说明：

(1) <类型名称>是子界类型的名称。

(2) <常量 1>是子界类型的下界，<常量 2>是子界类型的上界，子界类型的上下界应该属于相同的有序类型，如：同为整型或同为字符型等。

(3) 子界的上界大于或者等于下界。

例如：

```
type
  month=1..12;
  weekday= (sun,mon,tue,wed,thu,fri,sat);
  workday=mon..fri;
```

2. 子界类型变量的定义

子界类型的变量定义必须遵循先定义类型再定义变量的规定。例如：

```
type
  month=1..12;
  weekday= (sun,mon,tue,wed,thu,fri,sat);
  workday=mon..fri;
var
  m1,m2:month;
  w1,w2:workday;
```

也可以将类型定义和变量定义放在一起，但是不主张这样做。例如：

```
var
  m1,m2:1..12;
```

5.3 集合类型

集合是指具有相同性质且可以区分的对象全体，构成集合的元素称之为集合的元素。例如 100 以内的全体素数、一个班的全体男生、26 个大写的英文字母等。集合具有以下性质。

(1) 集合中的元素是相异的。例如，集合[1,2,3,4,2]就应该写成[1,2,3,4]。

(2) 集合中的元素是没有顺序的。例如 [1,2,3,4]和[2,4,1,3]表示同一个集合。

(3) 集合中的元素不能超过 256 个。

5.3.1 集合类型的定义

1. 集合类型定义的格式

集合类型定义的格式如下：

```
type
  <类型名称>=set of<基类型>;
```

说明：

(1) <类型名称>是用户所定义的集合类型的名称。

(2) <基类型>是集合中元素的类型，可以是字符、布尔、枚举、子界等类型，不能是整型、实型以及其他自定义类型。

例如：

```
type
  days=set of 28..31;
  ch=set of 'A'.. 'Z';
  weekday=(sun,mon,tue,wed,thu,fri,sat);
  workday=set of weekday;
```

2. 集合类型变量的定义

先定义集合类型，再定义集合类型的变量。例如：

```
type
  days=set of 28..31;
  ch=set of 'A'.. 'Z';
  weekday=(sun,mon,tue,wed,thu,fri,sat);
  workday=set of weekday;
var
  ch1,ch2:ch;
  wo1,wo2:workday;
  d1,d2:days;
```

还可以将集合类型定义和集合类型变量定义放置在一起，但是不主张这样做。例如：

```
var
  d1,d2:set of 28..31;
  ch1,ch2:set of 'A'.. 'Z';
  w1,w2:set of(sun,mon,tue,wed,thu,fri,sat);
```

5.3.2　集合类型的取值和运算

1. 集合类型的取值

集合的取值称之为集合的值。例如：

```
type
  weekday=(sun,mon,tue,wed,thu,fri,sat);
  workday=set of weekday;
var
  wo1,wo2:workday;
```

wo1 的值可以是[]、[sun]、[sun,mon]等。

说明：

(1) 如果一个集合的基类型有 n 个值，则集合变量的取值有 2^n 个。

(2) []是空集合。

(3) 集合元素连续出现,可以写成子界形式,例如[sun..wed,sat]和[sun,mon,tue,wed,sat]表示同一个集合。

2. 集合的运算

集合的运算有集合间的交、差、并运算和集合间的关系运算两种类型。前者得到的是集合类型,或者得到的是布尔类型。

(1) 集合的并。设有集合 A 和 B,A 集合和 B 集合的并表示为 A+B 或者 B+A,A+B 的集合元素是由两个集合中的所有元素组成且不允许重复。例如:[1,2,4]+[1,3,4,5]的值为[1,2,3,4,5]。

(2) 集合的交运算。设有 A 和 B 两个集合,A 和 B 集合的交表示为 A*B 或者 B*A,A*B 的集合元素是 A 和 B 中所有的公共的元素。例如,[1,2,3]*[2,4,5,1]的值为[1,2]。

(3) 集合的差。设有 A 和 B 两个集合,A 和 B 集合的差表示为 A-B,A-B 的集合元素是所有属于 A 集合而不属于 B 集合的元素。例如,[1,2,3]-[3,4,5]的值为[1,2]。

(4) 相等运算,判断两个集合是否相等。例如,[1,2,3]=[3,1,2]的值是 true。

(5) 不相等运算,判断两个集合是否不相等。例如,[1,2,3]<>[2,3]的值是 true;[1,2,3]<>[2,3,1]的值是 false。

(6) 包含运算,判断前面的集合是否包含后面的集合。例如,[1,2,3]>=[1,2]的值是 true;[1,2,3]>=[]的值是 true。

(7) 被包含运算,判断前面的集合是否被包含在后面的集合中。例如,[1,2,3]<=[1,2,3]的值为 true。

(8) 元素与集合的运算,判断元素是否在集合中。例如,2 in [1,2,3]的值为 true;2 in []的值为 false。

【例 5-2】 用"筛法"求 1～250 以内的所有素数。

所谓"筛"实际上就是集合,"筛法"求素数的方法如下。

(1) 首先把 1～100 放置在一个集合中,从集合中去掉最小的元素 1。

(2) 将集合中 2 的倍数去掉(2 本身除外)。

(3) 将集合中 3 的倍数去掉(3 本身除外)。

(4) 用类似的方法继续进行……,将 250 的所有倍数去掉(250 除外)。

说明:实际上经过前面删除集合元素的操作,后面根本用不着删除 250 的倍数。

设计步骤如下:

(1) 设计界面,如图 5-3 所示。

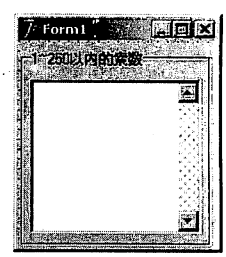


图 5-3 程序界面

(2) 编写事件过程 FormCreate 如下:

```
type
 num=set of 1..250;

var
 sushu:num;
```

```
procedure TForm1.FormCreate(Sender: TObject);

 var i,m:integer; s:string;

begin
 s:='';
 sushu:=[1..250];
 sushu:=[1..250]-[1];
 for i:=2 to 250 do
 begin
  m:=2*i;                          // i 乘以 2 是为了不去掉 i 本身
  repeat
   sushu:=sushu-[m];               //去掉 i 的倍数(i 本身不去掉,因为它是素数)
   m:=m+i;                         //m 始终是 i 的倍数
  until m>=250;
 end;
 for i:=1 to 250 do
 if i in sushu                     //如果 i 在 sushu 中,则 i 是素数
 then s:=s+inttostr(i)+'  ';
 memo1.Lines.Add(s);               //素数输出至 memo1 多行编辑框
end;
```

(3) 运行结果,如图 5-4 所示。

【例 5-3】 输入一行字符,求这行字符中英文字母、数字、空格和其他字符的个数。

分析:可以事先定义 3 个集合,用于保存英文字母、数字和空格。然后将输入的字符中的每个字符与这 3 个集合比较,如果该字符在某个集合中,则该字符就是这种类型的字符。

步骤如下:

(1) 设计界面如图 5-5 所示。

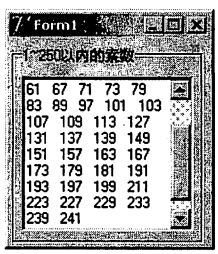


图 5-4　程序运行结果

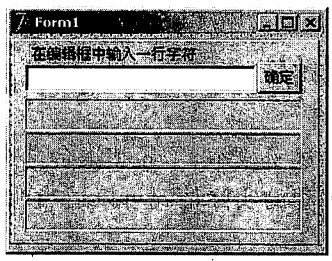


图 5-5　程序界面

(2) 编写 Button1 的 OnClick 事件过程如下:

```
type
 yingwen=set of #65..#122;
 shuzi=set of '0'..'9';
 kongge=set of ' '..' ';
procedure TForm1.Button1Click(Sender: TObject);
```

```
var yw,sz,kg,qt,i,n:integer; s:string;
    yw1:yingwen;sz1:shuzi;kg1:kongge;

begin
 yw1:=['a'..'z','A'..'Z'];
 sz1:=['0'..'9'];
 kg1:=[' '];
 yw:=0;sz:=0;kg:=0;qt:=0;
 n:=length(edit1.Text);
 s:=edit1.Text;
 for i:=1 to n do
  begin
   if s[i] in yw1 then yw:=yw+1;
   if s[i] in sz1 then sz:=sz+1;
   if s[i] in kg1 then kg:=kg+1;
  end;
 qt:=n-yw-sz-kg;
 panel2.Caption:='英文字母有'+inttostr(yw)+'个';
 panel3.Caption:='数字字符有'+inttostr(sz)+'个';
 panel4.Caption:='空格字符有'+inttostr(kg)+'个';
 panel5.Caption:='其他字符有'+inttostr(qt)+'个';
end;
```

(3) 运行程序，界面如图 5-6 所示。

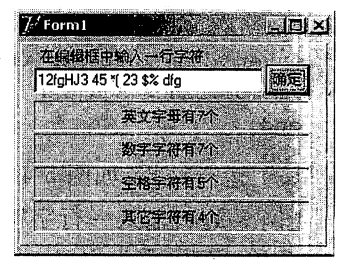


图 5-6 程序运行结果

说明：

(1) 千万要注意 s 和 s[i]是不一样的，前者是字符串，后者是字符，若混淆将会出现类型错误。

(2) 上面把集合类型 yingwen 定义为“yingwen＝set of ＃65..＃122;”，这里面包含了所有的大小写英文字母，以及少量的其他字符。yingwen 类型的集合变量 yw1 的初始值为 ['a'..'z','A'..'Z']，该初始集合中不再含有其他字符。

5.4 小结

本章讲述了枚举、子界和集合。本章相对其他章节来说，显得不是非常重要。教师根据课时和进程的实际情况可以适当地进行删减。

习题

1. 以下正确的枚举类型定义是：_____。

   A. type vowel= ('a','e','u','o','i');

   B. type num= (1..3,4..9);

   C. type color= (red,green,blue,white);

   D. type realtype= (3.6,7.6,5.3);

2. 若要用枚举类型表示加、减、乘、除运算，下列正确定义形式为：_____。

A. type op=(+,-,*,/);

B. type op=('+','-','*','/');

C. type op=(a+,b-,c*,d/);

D. type op=(add,sub,mul,divi);

3. 设有如下类型定义和变量声明：

```
colors=(red,green,yellow,white);
var
 color1,color2:colors;
```

判断下列语句是否正确？为什么？

(1) colors:= red;　　(2) color1= yellow;

(3) color2:= color1　　(4) color1:= 'yellow';

4. 设有如下类型定义和变量声明：

```
type
   weekday=(sun,mon,tue,wed,thu,fri,sat);
var
  w1,w2:weekday;
```

执行 w1:=mon;w2:=sat;后，指出下面的函数值。

(1) pred(mon)　　(2) pred(w1)

(3) ord(tue)　　(4) succ(thu)

(5) ord(succ(thu))　　(6) ord(pred(succ(w1)))

5. 下列子界类型定义正确的是：_____。

A. type num= 1.0..100.0;　　B. type num= 1..100;

C. type num= a..f;　　D. type num= 'Z'..'A';

6. 集合运算[1,2,3]+[1,3,4,2,5]的值是：_____。

A. [1,2,3,1,3,4,2]　　B. [1,2,3,4]

C. [1,3,4,2,5]　　D. [1,2,3,1,2,3,4,5]

7. 集合表达式['m','n','b']*['a','b','c']的值为：_____。

A. 'm'　B. 'n'　C. 'b'　D. 'a'　E. 'c'

8. 下列表达式中哪一个的值是空集？_____。

A. ['a'..'z']-['z','a','T']　　B. ['a'..'z']-['A'..'Z']

C. ['1'..'9']*['a'..'z']　　D. [1,2,3]*[2,3,4]

9. 下列表达式中哪一个是非法表达式？_____。

A. ['a'] in ['a'..'z']　　B. ['a']<=['a'..'z']

C. [1,2]+[3,4]*[4,5]>=[1,2,4]　　D. 12 in [1..10]

10. 将表达式

```
(ch='a') or (ch='o') or (ch='u') or (ch='e') or (ch='i')
```

写成集合类型的表达式。

11. 口袋中有红、黄、蓝、绿、黑 5 种颜色的球，每次从袋中取出 3 只球。编写程序求有多少种取法。如图 5-7 所示。

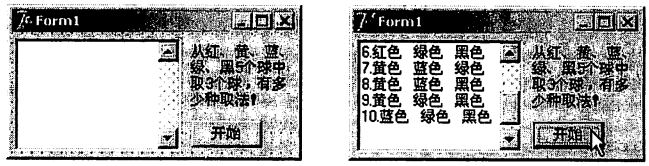


图 5-7 从 5 个不同颜色的球中取 3 个球的取法问题

12. 输入某年月，判断该年某月有多少天，该月是四季中的哪个季节。界面如图 5-8 所示。

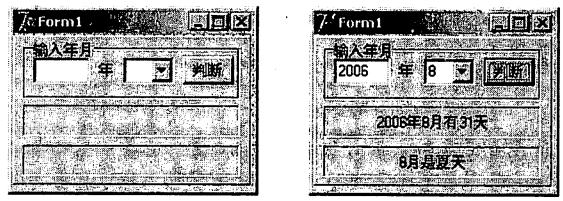


图 5-8 判断某年某月有多少天，是哪个季节

第 6 章

数组与记录

数组是一些具有相同类型的元素按顺序组成的序列。数组中的每一个元素不仅类型相同，而且变量的名称也相同，它们通过下标来相互区别，数组的下标是连续的，这些数组元素在内存中也是连续存放的。在现实编程过程中，往往需要将多个分量的数据放在一起进行操作，这时就可以使用所谓的记录来完成此类操作。

6.1 数组类型

在 Delphi 中，数组在内存中的存储不再局限于静态的，可以采用动态方式为数组分配存储单元。因此，Delphi 中的数组可以分为动态数组和静态数组。

6.1.1 静态数组

所谓静态数组是指在定义的时候就分配存储单元，确定数组元素的个数和类型的数组。

1. 一维静态数组

数组根据下标的个数分为一维、二维和多维数组。一维数组的下标只有 1 个，其声明的格式为：

```
type
  <类型标识符>=array[<下标类型>]of<基类型>;
```

说明：

(1) ＜类型标识符＞是用户定义的数组类型的名称。

(2) ＜下标类型＞必须是整数类型、字符类型、布尔类型、子界类型和枚举类型，也就是顺序类型。

(3) ＜基类型＞就是元素的类型，可以是任何类型，数组的每个元素的类型是一致的。

(4) 数组元素的个数就是下标的个数。

例如：

```
type
  xuehao=1..50;
```

```
  score=array[xuehao]of integer;
  acolor=(read,blue,green);
  yanse=array[acolor]of integer;
```

数组类型变量的定义，遵循先定义类型后定义变量的原则。例如：

```
type
  xuehao=1..50;
  score=array[xuehao]of integer;
  acolor=(red,blue,green);
  yanse=array[acolor]of integer;
var
  s1,s2:score;
  c1,c2:yanse;
```

也可以数组类型和数组元素一起定义，这样更加简洁，例如：

```
var
  a1,a2:array[1..30]of real;
```

数组的类型和数组的变量定义之后，就可以访问数组元素。函数 low 可以得到数组的最小下标，函数 high 可以得到数组的最大下标，函数 length 可以得到数组的元素个数。

例如：

low(s1)和 low(score)表示数组 score 下标的最小值，也就是 1。参数可以是变量和类型。

high(yanse)和 high(c1)表示数组 yanse 的最大下标，也就是 green。参数可以是变量和类型。

length(a1)的值是 30，表示数组元素的个数。length 的参数只能是变量，不能是类型。

数组类型和数组变量定义之后，就可以通过数组的变量名称和下标引用数组。例如：

```
a[1]:=a[1]+100.4;
c1[green]:=90;
s1[50]:=89;
```

【例 6-1】 随机产生 10 个 100 以内的正整数，求这 10 个数中最大的数，以及它的下标。

分析：因为这 10 个数的类型是一样的，而且定义这么多变量也不方便，可以使用数组类型来解决问题。

设计步骤如下：

(1) 添加组件，设置属性，界面如图 6-1 所示。

(2) 编写程序如下：

```
var
 a:array[1..10]of integer;

procedure TForm1.Button1Click(Sender: TObject);

 var max,i,p:integer;
```

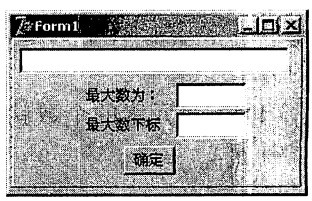


图 6-1 程序界面

```
begin
 edit1.Text:='';
 randomize;                                          //随机过程
 for i:=1 to 10 do
  begin
   a[i]:=round(random(100));                         //随机函数产生 10 个随机数
   edit1.Text:=edit1.Text+inttostr(a[i])+'  ';       //随机数显示在编辑框中
  end;
 max:=a[1];p:=1;                    //假设第一个元素最大,记下该数及其下标
 for i:=2 to 10 do                  //循环求最大
  if a[i]>max then
  begin
   max:=a[i];
  p:=i;                             //最大数的下标
  end;
  edit2.Text:=inttostr(max);        //输出到 edit2 和 edit3
 edit3.Text:=inttostr(p);
end;
```

(3) 运行程序,如图 6-2 所示。

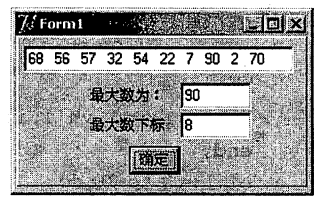


图 6-2　程序运行结果

2. 二维静态数组

所谓二维数组就是有两个下标的数组。可以理解为一个一维数组的每个元素又是一个一维数组。也就是：

```
type
  <数组类型名>=array[<下标类型 1>]of array [<下标类型 2>]of <元素类型>;
```

一般来说,习惯把二维数组定义为如下形式：

```
type
  <数组类型名>=array[<下标类型 1>,<下标类型 2>]of <元素类型>;
```

例如：

```
type
  aa=array[1..5,2..8]of integer;
var
  a1,a2:aa;
```

说明：上面定义的数组类型 aa 有两个下标,数组类型 aa 共有(5－1＋1)×(8－2＋1)＝35 个元素,每个元素都是整型的。

二维数组的用法与一维数组相同。

【例 6-2】　打印杨辉三角形,要求输出杨辉三角形的前 10 行。

```
1
1  1
1  2  1
1  3  3  1
1  4  6  4  1
……
```

分析：可以使用二维数组来输出杨辉三角形的元素。分析可知，杨辉三角形的第1列元素都是1，对角线元素都是1(不考虑三角形之外的元素)，其他元素是它正上面的元素和左上角的元素之和。因此，只要初始化那些值为1的元素，就可以根据这些元素求出其他元素的值。最后输出那些值不为0的元素即可。

设计步骤如下：

(1) 添加组件，设置属性，程序界面如图6-3所示。

(2) 编写程序如下：

```
var
 yh:array[1..10,1..10]of integer;

procedure TForm1.FormCreate(Sender: TObject);

 var i,j:integer;s:string;

begin
 for i:=1 to 10 do
  begin
   yh[i,1]:=1;yh[i,i]:=1;                    //第1列和对角线元素为1
  end;
 for i:=2 to 10 do
  for j:=2 to i do
   yh[i,j]:=yh[i-1,j-1]+yh[i-1,j];  //其他元素等于它的正上方元素和左上角元素之和
 for i:=1 to 10 do
  begin
   s:='';
   for j:=1 to i do s:=s+inttostr(yh[i,j])+'  ';
   memo1.Lines.Add(s);
  end;
end;
```

(3) 运行程序，结果如图6-4所示。

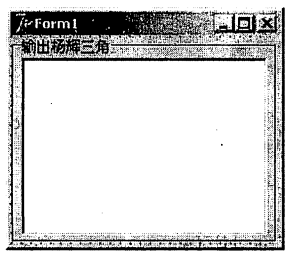


图6-3 程序界面

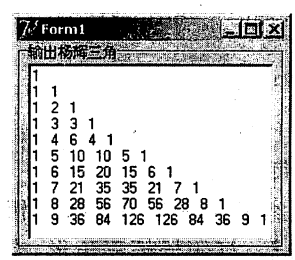


图6-4 打印出杨辉三角形前10行

【例6-3】 定义一个二维数组，求班上学生(不超过60个学生)4门功课的平均分。

分析：将20个学生的成绩存放到一个二维数组中，第一维下标相同的元素存放着同一门功课的成绩，第二维下标相同的元素存放着同一个学生的成绩。求每门功课的成绩只需要求第一维下标相同元素的平均值即可。

设计步骤如下：

(1) 在窗体上添加若干标签、2 个按钮、4 个单行编辑框和 1 个多行编辑框、1 个面板 panel。界面如图 6-5 所示。

图 6-5　程序界面

(2) 定义类型和变量如下：

```
type chengji=array[1..4,1..60]of real;
var
  cj:chengji;count:integer;
```

说明：用于表示 60 个学生的 4 门功课成绩，实际上可能没有 60 个学生，因此定义变量 count 表示学生的实际人数。

(3) 定义 function，用于计算每门功课的平均成绩。

```
function aver(cj1:chengji;m,n:integer):real;
  var i:integer;sum:real;
begin
 sum:=0;
 for i:=1 to n do
  sum:=sum+cj1[m][i];
 aver:=sum/n
end;
```

说明：aver 函数的第 1 个参数是数组类型，是第(1)步中定义的数组类型，第 2 个参数表示某门功课，第 3 个参数是学生实际人数。调用的时候，对应的实际参数可以是全局变量 count。

(4) 编写 Button1 的 OnClick 事件过程，用于输入成绩。如下：

```
procedure TForm1.Button1Click(Sender: TObject);

 var st:string;

begin
 count:=count+1;                    //每输入一个人的成绩，表示人数的 count 变量要加 1
 cj[1][count]:=strtofloat(edit1.Text);        //4 门功课的成绩保存到数组中
 cj[2][count]:=strtofloat(edit2.Text);
 cj[3][count]:=strtofloat(edit3.Text);
 cj[4][count]:=strtofloat(edit4.Text);
 st:=' '+edit1.Text+'    '+edit2.Text+'    '+edit3.Text+'    '+edit4.Text;
 memo1.Lines.Add(st);                        //成绩添加到 memo 多行编辑框
 edit1.Text:='';
 edit2.Text:='';
 edit3.Text:='';
 edit4.Text:='';
 edit1.SetFocus;
end;
```

(5) 编写 Button2 的 OnClick 事件过程，求每门课程的平均成绩。如下：

```
procedure TForm1.Button2Click(Sender: TObject);

 var delphi,wljs,vfp,sjk:real;

begin
  delphi:=aver(cj,1,count);                          //求 Delphi 的平均成绩
  wljs:=aver(cj,2,count);                            //求网络技术的平均成绩
  vfp:=aver(cj,3,count);                             //求 vfp 的平均成绩
  sjk:=aver(cj,4,count);                             //求数据库的平均成绩
  panel1.Caption:='Delphi:'+format('% f',[delphi]);  //显示平均成绩
  panel1.Caption:=panel1.Caption+',网络技术:'+format('%f',[wljs]);
  panel1.Caption:=panel1.Caption+',vfp:'+format('% f',[vfp]);
  panel1.Caption:=panel1.Caption+',数据库:'+format('% f',[sjk]);
end;
```

(6) 为了防止输入非法字符，编写 Edit1 的 KeyPress 事件过程。Edit2、Edit3 和 Edit4 的 KeyPress 事件过程共享 Edit1 的事件过程。代码如下：

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
 //禁止输入数字、小数点、退格键之外的字符，#8 是退格键，#46 是小数点
 if not (((key>=#48) and (key<=#57))or (key=#8)or (key=#46))
 then key:=#0;
end;
```

(7) 运行程序，界面如图 6-6 所示。

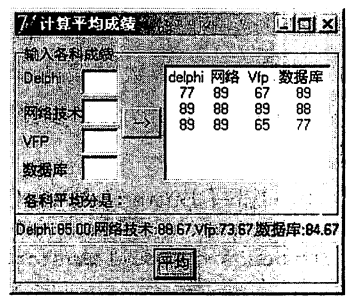


图 6-6 求平均成绩运行结果

6.1.2 动态数组

静态数组在定义数组类型的时候，声明了数组的大小。实际上在定义数组类型的时候可以不声明数组的大小，而在程序中根据需要申请分配内存空间，这种数组就称之为动态数组。

1. 一维动态数组

一维动态数组的定义格式为：

```
type
  <数组类型名称>=array of <基类型>
```

或者直接将类型和变量定义在一起：

```
var
  <变量名>:array of <基类型>
```

动态数组的定义中没有给出数组的大小，因此数组的大小可以在程序中动态改变。在 Delphi 中使用 Setlength 来明确动态数组的大小。

```
Var
  a:array of integer;
```

```
begin
 ...
  setlength(a,20);
  ...
end;
```

说明：语句 setlength(a,20)为 a 数组声明 20 个数组元素。下标从 0 到 19，即 a[0]～a[19]。

2. 多维动态数组

声明多维动态数组与声明多维数组类似，以二维数组为例，语法格式为：

```
type
  <数组类型名称>=array of array of <基类型>
```

或者直接将类型和变量定义在一起，如下。

```
var
  <变量名>:array of array of <基类型>
```

例如：

```
Var
  a:array of array of integer;

begin
 ...
  setlength(a,4,6);
  ...
end;
```

说明：数组 a 共有 24 个元素，分别是 a[0,0],a[0,1],…,a[0,5],…,a[3,5]。

6.1.3　字符串类型

在第 2 章中已经讲述了字符串的一些基本的知识，本章将从数组的角度讲述字符串。字符串实际上是以字符为数组元素的一维数组。

可以这样定义字符串：

```
type
  <字符串类型名>=string;
var
  <字符串变量>:<字符串类型名>;
```

或者：

```
<字符串变量>:string;
```

在 Delphi 中还允许在定义字符串的时候声明字符串中字符的个数，格式为：

```
type
```

```
  <字符串类型名>=string[<长度>];
var
  <字符串变量名>:<字符串类型名>;
```

或者：

```
var
  <字符串变量名>:string[<长度>];
```

例如：

```
type
  xingming=string[10];
var
  xm1:xingming;
```

xm1 的长度最多是 10 个字符，可以给变量 xm1 赋值，例如 xm1 := 'li si'，则 xm1 分配的内存空间如图 6-7 所示。

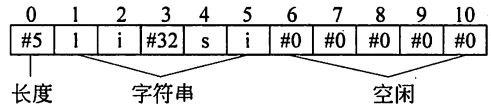


图 6-7 字符串存储示意图

其中，s[0]=#5 表示字符串的长度，s[1]='1'表示字符串的第 1 个字符，s[2]='i'表示字符串的第 2 个字符，s[3]=#32 是空格表示字符串的第 3 个字符，…，s[6]～s[10]都是空字符。利用 ord(s[0])可以求出字符串的长度。

6.2 记录类型

在现实中，往往需要将一些数据集中在一起，作为一个整体进行操作。例如，表示学生的信息包括学号、姓名、语文。Delphi 提供了一个称之为记录的数据类型可以解决这样的问题，而学号、姓名和语文可以看成集合中的 3 个字段。

6.2.1 记录类型的定义

记录类型的定义格式为：

```
type
  <记录类型名>=record
     <字段 1>:<类型 1>;
     <字段 2>:<类型 2>;
     ⋮
     <字段 n>:<类型 n>;
end;
```

说明：记录类型名是用户所定义的记录类型的名称，后面的 record 表示定义的是记录，在记录定义完之后一定要加 end 表示记录定义结束。在记录中定义了一些字段，这些

字段和以前普通变量的定义方法一样。

例如：

```
type
 student=record
  xh:string[2];
  xm:string[8];
  yw:integer;
end;
                                    //student 含有学号、姓名、语文 3 个字段
var
 st:array[1..20]of student;         //班上学生不到 20 人
 s1,s2:student;
```

上面定义的记录类型 student 有 3 个字段，分别是 xh(学号)、xm(姓名)、yw(语文)。记录的字段可以是各种类型，甚至可以是记录类型。st 是数组类型，其基类型是 student 记录类型，表示一个班多个学生的信息，s1、s2 也是 student 记录类型。

也可以将记录类型的定义和记录变量的定义放在一起，但由于记录类型的定义比较复杂，本身容易出错，因此主张将记录类型和记录变量的定义分开进行。

6.2.2　记录类型的访问与 with 语句

在程序设计中，记录类型和记录变量定义好之后，就可以访问记录了。记录的访问包括访问记录的字段以及整个记录。

1. 记录的访问

记录变量定义之后，就可以访问记录的每个字段了，访问记录字段的格式为：

```
<记录变量名>.<字段名>;
```

也可以将整个记录变量看成是一个整体，对它进行整体访问，这种访问方法和一般变量的访问方法是一样的。

请看下面的程序：

```
type
 student=record
  xh:string[2];
  xm:string[8];
  yw:integer;
end;

var
 st:array[1..20]of student;
 s1,s2:student;

begin
  ...
 st[3].xm:='张志远';
 st[2].yw:=97;
```

```
  s1:=st[5];
  st[5]:=st[4];
  st[1]:=s2;
  ...
end;
```

说明：

(1) 对记录访问，可以访问记录的某些字段，例如语句 st[3].xm :='张志远'和语句 st[2].yw :=97 就分别访问了 st[3]的 xm 字段以及 st[2]的 yw 字段。

(2) 对记录的访问，还可以对整个记录变量进行访问，例如语句 st[5] :=st[4]和语句 st[1] :=s2 以及语句 s1 :=st[5]等。

【例 6-4】 输入班上学生的信息，学生信息包括学号、姓名和语文。要求按照学号顺序输入信息，然后，按照语文分数从高到低的顺序排列，将排序结果输出到列表框。

分析：由于学生信息是由 3 部分构成的，因此可以考虑使用记录类型；另外，一个班有多个学生可以使用记录类型的数组。

根据分析，设计步骤如下：

(1) 添加组件，在窗体中添加编辑框 3 个、标签 3 个、GroupBox 1 个、列表框 1 个和 2 个命令按钮，并设置组件的属性，界面如图 6-8 所示。

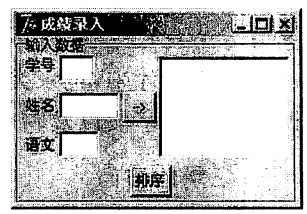


图 6-8 程序界面

(2) 定义记录类型 student 以及记录数组和变量 count。如下：

```
type
 student=record
  xh:string[2];
  xm:string[8];
  yw:integer;
end;                                  //student 含有学号、姓名、语文 3 个字段

var
 st:array[1..20]of student;           //班上学生不到 20 人
 count:integer;                       //用于表示班上学生实际人数
```

(3) 编写 Button1 的 OnClick 事件过程，用于输入学生的信息。代码如下：

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 count:=0;
end;

procedure TForm1.Button1Click(Sender: TObject);

 var s:string;

begin
 count:=count+1;                      //人数
 st[count].xh:=edit1.Text;            //输入信息，保存到数组元素
```

```
 st[count].xm:=edit2.Text;
 st[count].yw:=strtoint(edit3.Text);
 s:=st[count].xh+'  '+st[count].xm+'  '+inttostr(st[count].yw);
 listbox1.Items.Add(s);          //输入的信息显示在列表框 listbox1 中
 edit1.Text:='';
 edit2.Text:='';
 edit3.Text:='';
 edit1.SetFocus
end;
```

(4) 编写 Button2 的 OnClick 事件过程,对记录进行排序。程序如下:

```
procedure TForm1.Button2Click(Sender: TObject);

 var t:student;i,j:integer;s:string;

begin
 for i:=1 to count-1 do          //按照从高到低排序
  for j:=1 to count-i do
   if st[j].yw<st[j+1].yw then
    begin
     t:=st[j];
     st[j]:=st[j+1];
     st[j+1]:=t;
    end;
 listbox1.Items.Clear;
 for i:=1 to count do            //排序后重新显示在列表框 listbox1 中
  begin
   s:=st[i].xh+' '+st[i].xm+' '+inttostr(st[i].yw);
   listbox1.Items.Add(s);
  end;
end;
```

(5) 运行程序,结果如图 6-9 所示。

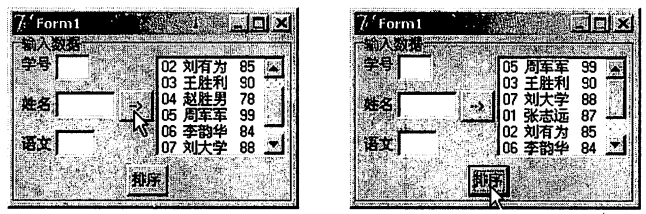


图 6-9　程序运行结果

2. With 语句

我们在前面讲到,如果要访问记录的字段,则需要加上记录变量这个前缀,如果使用到的字段较多,势必会给用户造成很大不便。幸好 Delphi 提供的 With 语句可以很好地解决这个问题。

With 语句的语法为:

With<变量名>do 语句;

如果后面的语句是单个语句，使用 With 语句没有什么意义，一般来说后面的语句是复合语句。例如：

```
type
 student1=record
  xuehao:string[2];
  xingming:string[8];
  yuwen:integer;
  xingbie:(male,female)
end;

var
 s1:student1;

with s1 do
 begin
  xingming:='zhangsan';
  yuwen:=90;
 end;
```

说明：上面表示的含义是 s1. xingming :='zhangsan'和 s1. yuwen :=90。

为了方便，Delphi 还允许 With 语句的嵌套。例如：

```
type
 student1=record
  xuehao:string[2];
  xingming:string[8];
  yuwen:integer;
  xingbie:(male,female)
 end;

 student2=record
  xuehao:string[2];
  xingming:string[8];
  yuwen:integer;
  zhuzhi:string[20];
 end;

var
 s1:student1;
 s2:student2;

with s2 do
  with s1 do
   begin
    xingming:='zhangsan';
    yuwen:=90;
   end;
```

说明：s1 和 s2 这两个变量中都有 xingming 和 yuwen 字段，with 之后的语句：

```
xingming:='zhangsan';
yuwen:=90;
```

表示的是 s1 的 xingming 和 yuwen，也就是说 s1 优先于 s2。上面的两个语句还可以写成：

```
with s2,s1 do
  //s2 在 s1 前面,s1 优先,因此下面语句中 xingming 和 yuwen 是属于 s1 变量的
  begin
   xingming:='zhangsan';
   yuwen:=90;
  end;
```

这种写法很容易产生歧义，因此建议采用第一种写法。在 Delphi 中用户还可以在记录中定义记录类型，这就是记录的嵌套，访问嵌套记录的时候，也可以使用嵌套的 With 语句，用法与上面一致。

6.3　小结

本章讲述了数组和记录。数组中讲述了静态数组和动态数组以及字符串类型。记录类型讲述了记录类型的定义以及记录的访问与 With 语句。数组是本章的重点，读者应该学会使用数组以及配合前面的循环来编写程序。

习题

1. 定义一个 m * n 的数组，随机产生这个数组中每个元素的值，求这个数组最大元素的值以及该元素的下标。

2. 定义一个记录数组，该记录数组含有学号、姓名、语文 3 个字段，编写程序实现添加记录、删除记录、查询记录的操作。界面如图 6-10 所示。

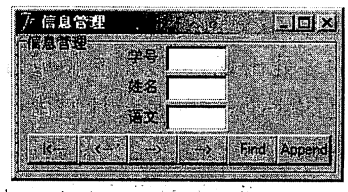


图 6-10　程序界面

3. 求一个二维数组的鞍点，所谓鞍点就是一个数在一行中是最大的同时在一列中又是最小的，也可能没有鞍点。

4. 定义一个 n * n 的二维数组，将这个二维数组的对角线元素赋值为 1，其他元素赋值为 0。

5. 定义一个 m * k 的矩阵和一个 k * n 的矩阵，将这两个矩阵相乘，得到一个 m * n 的矩阵，输出这个矩阵。

第 7 章

过程与函数

过程(procedure)与函数(function)是实现面向对象编程思想的重要手段。在 Delphi 中将各个功能模块做成函数或者过程的形式,供程序不同位置的调用,被调用的函数和过程也称为子程序。使用函数和过程不仅可以将一个大而复杂的问题简单化,而且还可以使程序更加清晰、条理。

过程与函数的区别在于过程没有返回值,而函数有返回值。一般来说,定义过程的目的是为了实现某个功能、执行某个操作,而定义函数的目的在于计算得到某个值。

7.1 过程

在 Delphi 中过程分为系统标准过程和自定义过程。系统标准过程是系统内部定义的,无须定义即可直接调用,如第 2 章所讲的 EncodeDate 过程,用户无须定义它,即可直接调用实现将年、月、日合并成为日期类型的功能。自定义过程也分为两类,即事件过程和通用过程。

事件过程的执行有两种方式,一种是事件驱动,即一个事件发生了,系统自动执行相应的事件过程,还有一种就是调用并执行事件过程。事件过程与对象有关,是 Delphi 应用程序中最重要的过程。

通用过程不依附于对象,一般用来执行某些操作,完成某个功能。有些功能代码在程序的多处出现,可以把它做成过程,在多处调用它,这样可以使程序清晰、简洁,避免书写重复的代码,这就是设计通用过程的理由。为了加深对过程以及过程调用的理解,需要重新回顾事件过程的定义和调用。

7.1.1 事件过程的创建与调用

在对象监视器中选择某个对象(或者在窗体中单击该对象),然后在对象监视器的 Event 中选择相应的事件并双击右边的空白区域,Delphi 将自动产生一个默认的事件过程。事件过程的名称是对象名加上事件类型名,如 Button1 的 OnClick 事件对应的事件过程就是 ButtonClick。

当一个事件发生后,Delphi 会自动执行相应的事件过程。当然,一个事件过程也可以被多个事件甚至多个对象共享。方法是,首先为某个事件创建事件过程,然后在创建其

他事件的事件过程时，从事件名称右边的下拉组合框中选择已经建立的事件过程名称即可。

共享事件的好处是，当共享事件过程的多个事件中任何一个事件发生时都会执行事件过程。有时候很多相似组件共享事件过程可以减少代码的编写。

【例 7-1】 在窗体上添加 9 个 Image 组件 Image1～Image9，添加 Shape 组件若干，用于做 Image 组件的边框。设计界面如图 7-1 所示。

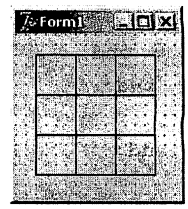


图 7-1　程序界面

本程序是两人对弈，双方在 9 格的棋盘上先后布棋，单击 Image 显示不同的图片（圆或叉），哪方的三个图片先成一条直线，哪方获胜。

步骤如下：

(1) 将 Image1～Image9 的 Tag 属性分别设置为 1～9。

(2) 定义全局变量：

```
n:integer;
flag:array[1..9]of integer;
```

说明：

① n 用于控制步数，奇数步和偶数步分别布“叉”棋和“圆”棋。

② flag 数组用于判断某个位置的棋的性质。

flag[x]= －1：表示 x 位置没有布棋；

flag[x]=0：表示 x 位置布“圆”棋；

flag[x]=1：表示 x 位置布“叉”棋。

(3) 在 TForm1.FormCreate 事件过程中将所有位置的棋子置为没有布棋（flag[i] := －1，i 从 1 到 9），将 n 置初值为 0，以后每布一颗棋子 n 就加 1。代码如下：

```
procedure TForm1.FormCreate(Sender: TObject);

 var i:integer;

begin
 n:=0;            //还没有布棋子
 for i:=1 to 9 do
  flag[i]:=-1;  //9 个 Image 均没有布棋子
end;
```

(4) 在对象监视器中选定 9 个 Image 组件，在 Event 中选择 OnClick 事件并双击空白区域，编写如下代码：

```
procedure TForm1.Image9Click(Sender: TObject);
begin

 if (flag[(sender as timage).Tag]=-1) then                    //如果该位置没有布棋子
 begin
  n:=n+1;
  if n mod 2=0 then                                           //偶数布“圆”棋子
```

```
    begin
     (sender as timage).picture.loadfromfile('yuan.bmp');
     flag[(sender as timage).Tag]:=0;                    //布子后,做标记
     end
    else                                                 //奇数布"叉"棋子
     begin
      (sender as timage).picture.loadfromfile('cha.bmp');
      flag[(sender as timage).Tag]:=1;                   //布子后,做标记
     end;
  end;
  ...                                                    //省略判断胜负的代码
end;
```

(5) 运行程序,界面如图 7-2 所示。

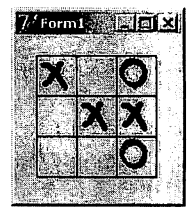


图 7-2 程序运行结果

说明:

(1) 本程序 9 个 Image 组件的 OnClick 事件都共用同一个事件过程,这样显然节省代码的编写工作量。

(2) 本程序中被单击的组件表示为(sender as timage),因此可以很方便地为它设置 picture 属性。

(3) 全局变量 n 用于控制布棋的奇偶步数,根据 n 是否能被 2 整除可知目前应该布"叉"棋还是布"圆"棋。

(4) Tag 的作用有两个。一是用于区分不同位置的 Image 组件,这样可以方便设置该位置的棋子属性(是否有棋,有什么棋);二是用于判断胜负,例如:flag[(sender as timage).Tag](Tag 为 1、2、3 时)的值都是 0,显然说明"圆"棋获胜等。请读者自己完成判断获胜的代码。

(5) 上面是 9 个组件共用一个事件过程。当然,也可以只为 Image9 编写过程 TForm1.Image9Click。而在 Image1～Image8 的事件过程中调用 TForm1.Image9Click 事件过程。例如:

```
procedure TForm1.Image1Click(Sender: TObject);      // Image2~Image8 类似
begin
 Form1.Image9Click(Sender)
end;
```

还可以写成:

```
procedure TForm1.Image1Click(Sender: TObject);
begin
  Form1.Image9Click(form1.Image1)
end;
```

通过本例,读者应该对事件过程的创建与调用有一个较深的认识。

7.1.2 通用过程

通用过程要遵循先定义后使用的原则,只用经过定义的过程才能够被程序调用,过程是没有返回值的,因此过程的调用就是一个独立执行的语句。

1. 通用过程的定义

通用过程的定义语法形式为：

```
Procedure<过程名>[(<形参表>)];
  [局部声明];
begin
  <语句序列>;
end;
```

<形参表>的格式为：

```
[Const|Var ]<形参名>:类型
```

说明：

(1) 保留字 Procedure 是定义通用过程的首部，“过程名”遵循标识符的命名规则。

(2) <形参表>由若干个形式参数组成，形式参数的类型和个数都必须给定。同类型的形式参数之间用逗号隔开，不同类型的形式参数之间用分号隔开，如果没有形式参数，则括号可省略。例如：

```
m,n:integer;var gys:integer
```

(3) 局部声明部分可以声明局部变量、常量、类型等。

(4) 程序体以 begin 开始，以 end 结束，end 之后有分号。

2. 通用过程的调用

程序执行某个过程称为调用，调用的格式为：

```
过程名(实参表);
```

说明：

(1) 实参表中参数的个数和类型必须和形参一致，实参是按照在参数表中的先后顺序将参数值依次传递给形参的。

(2) 如果形参用前冠以 var，则相应的实参必须是变量，否则只需要是同类型的表达式即可。

3. 通用过程举例

【例 7-2】 编写一个过程，输出 m 行“*”，要求第一行输出 1 个“*”，以后每行比上一行多出一个“*”，要求使用控制台应用程序。

设计步骤如下：

(1) 选择 File | New | Other... 出现新建项目对话框，在新建项目对话框中选择 Console Application，出现代码编辑器。

(2) 在代码编辑器中输入程序如下：

```
program Project1;

{$APPTYPE CONSOLE}

uses
```

```
 SysUtils;
var n:integer;

procedure prin(m:integer);

 var i,j:integer;

begin
 for i:=1 to m do
  begin
   for j:=1 to i do
    write('*');
   writeln;
  end;
end;
begin
 write('输入行数:');
 readln(n);
 prin(n);
 readln;
end.
```

(3) 运行程序,输入行数 6,结果如图 7-3 所示。

说明:

(1) 本程序定义的变量 n 在主程序和过程里面都是有效的;定义的 i,j 仅在过程中有效,是局部变量。

(2) 形参 m 是整型,实参 n 也应该是整型。

【例 7-3】 编写通用过程,用于求两个数或三个数的最大公约数。界面如图 7-4 所示。

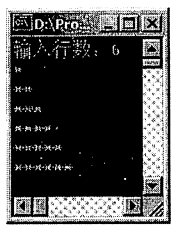


图 7-3 程序运行结果

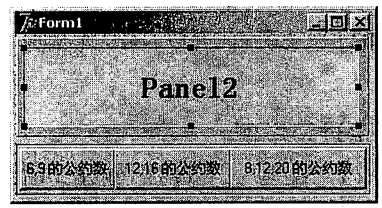


图 7-4 求公约数界面图

分析:由于求公约数在多处出现,因此最好把它定义为子程序,以便在多处调用,鉴于目前读者只学了过程,所以本例编写通用过程来解决问题。

步骤如下:

(1) 求两个数 m 和 n 的最大公约数有一种方法叫做辗转相除法,即:求 m 和 n 的最大公约数就是求 n 和 m mod n 的最大公约数,这样问题就变成了求 n 和 m mod n 的最大公约数了。将 n 看成新的 m,m mod n 看成新的 n 继续使用辗转相除法直到 n 为 0,此时的 m 就是所求的最大公约数。根据分析过程,gys 定义如下:

```
procedure gys(m,n:integer;var result:integer);
begin
 repeat
  result:=m;
  m:=n;
  n:=result mod n;
 until n=0;
 result:=m;
end;
```

说明：形式参数 result 必须定义为变量参数，以便将结果回传给实际参数，这点将在后面的章节中详细讲解。

(2) 编写 Button1 的 OnClick 事件过程如下：

```
procedure TForm1.Button1Click(Sender: TObject);

 var result1:integer;

begin
 gys(6,9,result1);
 panel2.Caption:='最大公约数为:'+inttostr(result1);     //Panel2 是小面板
end;
```

(3) 编写 Button2 的 OnClick 事件过程如下：

```
procedure TForm1.Button2Click(Sender: TObject);

 var result1:integer;

begin
 gys(12,16,result1);
 panel2.Caption:='最大公约数为:'+inttostr(result1);
end;
```

(4) 编写 Button3 的 OnClick 事件过程如下：

```
procedure TForm1.Button3Click(Sender: TObject);

 var result1,result2:integer;

begin
 gys(8,12,result1);              //带回的 result1 为 8 和 12 的最大公约数
 gys(result1,18,result2);        //求 result1 和 18 的最大公约数,结果由 result2 带回
 panel2.Caption:='最大公约数
 为:'+inttostr(result2);
end;
```

(5) 运行程序，结果如图 7-5 所示。

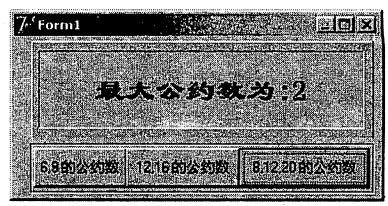


图 7-5　求最大公约数

说明：例 7-2 的过程用于控制台应用程序，而例 7-3 的过程用于 Windows 应用程序。另外，例 7-3 中的过程返回(抑或带回)一个结果，例 7-2 却进行某种

操作，因此活用 Delphi 过程将会为程序增色不少。

7.2 函数

和过程一样，函数也是子程序的一种形式。函数与过程的区别在于函数有返回值，过程没有返回值。Delphi 中的函数同样分为标准函数和自定义函数，标准函数如 sin、cos、exp、datetostr 等，标准函数无须定义，直接调用即可。本节讲述自定义函数，自定义函数要先定义后调用，调用的方法和标准函数一样。

7.2.1 函数的定义

自定义函数要遵循先定义后使用的原则，只用经过定义的函数才能够被程序调用，函数定义的格式是：

```
Function<函数名>[(<形参表>)]:<函数类型>;
  [局部声明];
begin
  <语句序列>;
end;
```

<形参表>的格式为：

```
[Const | Var ]<形参名>:类型
```

说明：

(1) 保留字 Function 是定义函数的首部，“函数名”遵循标识符的命名规则。

(2) <形参表>由若干个形式参数组成，形式参数的类型和个数都必须给定，同类型的形式参数之间用逗号隔开，不同类型的形式参数之间用分号隔开，如果没有形式参数，则括号可省略。例如：

```
m,n:integer;var gys:integer
```

(3) 局部声明部分可以声明局部变量、常量、类型等。

(4) 程序体以 begin 开始，以 end 结束，end 之后有分号。

(5) <函数类型>是函数返回值的类型。函数有返回值，其通过函数名带回，因此，在函数定义部分需要给函数名赋值。如果在函数体中不给函数名赋值，则函数的返回值就是默认值，数值型的默认值是 0，布尔类型是 False，字符类型是空字符等。

【例 7-4】 编写一个自定义函数，用于求整数 n 的阶乘。

```
function fac(n:integer):int64;          //返回值定义为 integer 容易溢出

 var m:int64;i:integer;

begin
 m:=1;
 for i:=1 to n do
  m:=m*i;
```

```
 fac:=m;
end;
```

7.2.2　函数的应用举例

【例 7-5】　求 100 以内的素数之和。

分析：因为本程序需要多处求素数，因此可以考虑定义一个函数用于判断某个数是否为素数。这个函数是布尔类型，返回值为 True 表示这个数是素数，返回值为 False 表示这个数不是素数。

步骤如下：

(1) 设计界面如图 7-6 所示。

(2) 设置函数如下：

```
function prime(n:integer):boolean;

 var i:integer;

begin
 prime:=true;
 for i:=2 to trunc(sqrt(n)) do
  if n mod i=0 then prime:=false;
end;
```

说明：首先假设 n 是素数，令 prime：＝true，然后在 2～$\sqrt{n}$之间寻找 n 的因子，如果找到 n 的因子则前面的假设是错误的，重新令 prime：＝false。如果找不到任何因子则 n 是素数，此时 prime 仍然保持为初始假设的 True 值。

(3) 编写 BitBtn1 的事件过程，如下：

```
procedure TForm1.BitBtn1Click(Sender: TObject);

 var s,i:integer;

begin
 s:=0;
 for i:=2 to 100 do
  if prime(i) then s:=s+i;
 panel2.Caption:='100 以内素数之和为'+inttostr(s);
end;
```

(4) 程序运行，界面如图 7-7 所示。

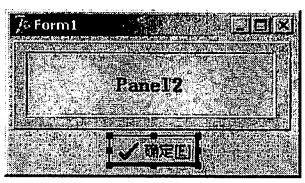


图 7-6　程序界面

图 7-7　程序运行结果

【例 7-6】 求两个数的最大公约数和最小公倍数。

分析：求两个数的最大公约数可以使用辗转相除法，而最小公倍数和最大公约数有这样的数学关系，即 m 和 n 的最小公倍数乘以它们的最大公约数刚好等于 m 和 n 的乘积。因此，只要求出最大公约数，就可以求出最小公倍数。因为二者特殊的数学关系，本例将求最大公约数和最小公倍数定义为一个函数。

设计步骤如下：

(1) 设计界面如图 7-8 所示。

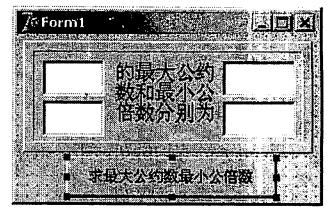


图 7-8 求公约数和公倍数的界面

(2) 编写函数的代码如下：

```
function divmul(m,n,flag:integer):integer;

 var t,a,b:integer;

begin
 a:= m;b:=n;                    //在 m 和 n 值改变之前,先将原值保留到 a 和 b 中备用
 repeat
  t:=m;
  m:=n;
  n:=t mod n;
 until n=0;
 if flag=1
 then divmul:=m                 //最大公约数
 else divmul:=a * b div m;      //最小公倍数
end;
```

说明：

① 由于求 m 和 n 的最小公倍数需要使用到最大公约数、m 和 n，所以先将 m 和 n 的值保留在 a 和 b 中，以便在求最小公倍数的过程中使用。

② 本自定义函数的参数中巧妙地定义了一个 flag，flag 值为 1 就是求最大公约数的标志，flag 值为其他值表示求最小公倍数。

(3) 编写 Button1 的事件过程如下：

```
procedure TForm1.Button1Click(Sender: TObject);

 var m,n:integer;

begin
 m:=strtoint(edit1.Text);
 n:=strtoint(edit2.Text);
 edit3.Text:=inttostr(divmul(m,n,1));          //edit3 中是最大公约数
 edit4.Text:=inttostr(divmul(m,n,0));          //edit4 中是最小公倍数
end;
```

(4) 运行程序，结果如图 7-9 所示。

说明：

(1) 例 7-5 巧妙地将函数结果定义为布尔类型，然后在程序中根据 prime(i)的值来决

定是否加到 s 中。

(2) 例 7-6 使用到参数 flag,根据参数 flag 值的不同,函数返回不同的值,这是该程序的巧妙之处。

上述两个例题各有千秋,读者应该举一反三、灵活运用。

(3) 需要强调的是函数的使用必须遵循先定义后使用的原则,也就是说函数定义的代码必须写在调用程序之前。

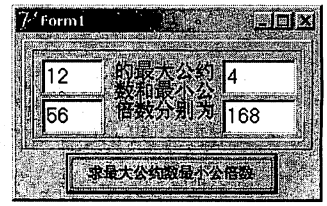


图 7-9　程序运行结果

7.3　参数传递

在定义函数或过程的时候所使用的参数称之为形式参数,简称形参;在调用函数或过程的时候所使用的参数称之为实际参数,简称实参。实参和形参必须一一对应,而且对应参数的类型也必须相同。参数传递是指调用程序按照先后顺序将实际参数一一传递给对应的形式参数。按照参数传递方式的不同,形式参数还可以分为变量参数、常量参数、值参数。

7.3.1　变量参数

在过程或者函数定义的时候,如果在形式参数前冠以保留字 var,则表示该参数是变量参数。变量参数值的传递方式是地址传递,所谓地址传递是指实际参数将自己的内存地址传递给形式参数,这样形式参数实际上和实际参数表示同一个存储单元。形式参数值的改变将会影响实际参数值。

例如,在例 7-3 中利用过程 procedure gys(m,n:integer;var result:integer)求公约数,在定义过程的时候形式参数 result 就是变参,而在如下过程中 result 的值就是最大公约数。

```
procedure gys(m,n:integer;var result:integer);
begin
 repeat
  result:=m;
  m:=n;
  n:= result mod n;
 until n=0;
 result:=m;
end;
```

上述变量参数的值通过地址传递给调用程序中的变量 result1,在 TForm1.ButtonClick 中 result1 和变量参数 result 就是一个存储单元。

```
procedure TForm1.Button1Click(Sender: TObject);

 var result1:integer;

begin
 gys(6,9,result1);
```

```
 panel2.Caption:='最大公约数为:'+inttostr(result1);
end;
```

说明：由于形式参数和实际参数表示同一个存储单元，因此实际参数必须是变量。

7.3.2 值参数和常量参数

在定义函数或者过程的时候，如果参数前被冠以保留字 const，那么该参数就是常量参数；如果参数前既没有保留字 var 也没有保留字 const，那么该参数就是值参数。

1. 值参数

值参数值的传递是按照值来传递的，即将实参的值传递给形参，形参和实参不是一个存储单元，形参值的改变不会影响实参。

2. 常量参数

无论实参是否是变量，常量参数在过程中其值不能改变。

7.3.3 默认参数

在定义函数或者过程的时候，可以给形式参数指定一个默认值，指定默认值的方法是：在形式参数后使用“=”，并给出一个具体的值。例如：

```
Procedure printline(m:integer;n:integer=5);
```

对于上面定义的过程，下面的两个调用语句是等价的。

```
prinline(a,5);
prinline(a);
```

Delphi 规定，在函数或者过程定义的时候，不能让前面的形式参数有默认值而后面的参数没有默认值。例如，下面这样定义是错误的。

```
Procedure printline(m:integer= 5;n:integer);
```

这是因为 Delphi 的参数传递是从前往后依次按照顺序传递的。

【例 7-7】 教职工信息录入系统。教职工的信息包括姓名、性别、民族。因为教职工中男职工多女职工少，汉族教职工多少数民族教职工少，因此，为提高数据录入速度，在定义过程的时候将男性、汉族作为默认值。

设计步骤如下：

(1) 在窗体上添加组件面板 Panel1，多行编辑框 Memo1，单行编辑框 Edit1、Edit2、Edit3，标签 Label1、Label2、Label3，按钮 Button1，设置属性。设计界面如图 7-10 所示。

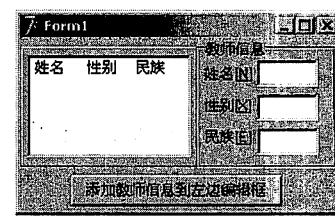


图 7-10 教职工信息输入窗口

(2) 编写自定义过程完成教职工信息添加，自定义过程如下：

```
procedure import(xm:string;xb:widestring='男';mz:string='汉族');
begin
 xm:=xm+'                                    //每个人的姓名长度可能不一样
```

```
 xm:=copy(xm,1,8);                                    //在姓名后加空格,补足 8 位
 xb:=xb+'  ';                                         //性别之后空两个小格
 form1.Memo1.Lines.Add(xm+'  '+xb+'  '+mz);
end;
```

说明：姓名是必须的；性别是默认的，默认值“男”；民族是默认的，默认值“汉族”。

(3) 编写 Button1 的事件过程如下：

```
procedure TForm1.Button1Click(Sender: TObject);

 var a:word;

begin
 if length(trim(edit1.Text))=0
 then a:=messagedlg('请输入姓名',mterror,[mbok],1) //姓名不可少
 else if length(trim(edit2.Text))=0                 //性别默认,此时民族也取默认值
    then import(edit1.Text)                         //仅有一个参数,是姓名
      else if length(trim(edit3.Text))=0            //性别不默认,但是民族取默认值
           then import(edit1.Text,edit2.Text)       //此时,有两个参数姓名和性别
           else import(edit1.Text,edit2.Text,edit3.Text);
             //三个编辑框都不空,三个参数,分别是姓名、性别、民族
 edit1.Text:='';
 edit2.Text:='';
 edit3.Text:='';
 edit1.SetFocus;
end;
```

说明：

① 男性和汉族是默认值。在输入时如果只输入姓名，则性别取默认值“男性”，此时即使输入民族也没有用，民族此时也取默认值“汉族”。

② 在输入信息的时候，如果输入了姓名、性别，若没有输入民族，此时民族取默认值“汉族”。

(4) 运行程序，结果如图 7-11 所示。

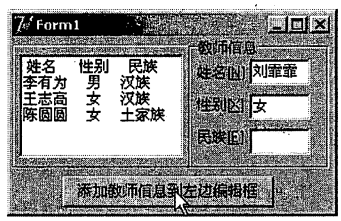


图 7-11　程序运行结果

7.4　子程序的嵌套与递归

在子程序(函数或者过程)中定义了另外的子程序(函数或者过程)，就称之为子程序(函数或者过程)的嵌套。在 Delphi 中子程序(函数或者过程)直接或者间接调用它自己就称之为递归，递归分为直接递归和间接递归。

7.4.1　子程序的嵌套

子程序中可以定义子程序，而小的子程序中还可以定义更小的子程序，这就是子程序的层数。Delphi 规定上级子程序必须完整地包含下级子程序，如图 7-12 所示，不允许局部包含，也就是如图 7-13 所示的交叉包含。

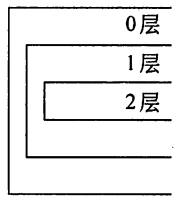


图 7-12 正确的嵌套关系

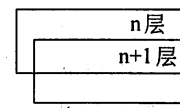


图 7-13 错误的嵌套关系

【例 7-8】 输入 m、n 的值求组合数 C_m^n。

分析：已知 $C_m^n=\frac{m!}{n!(m-n)!}$，因此要想求 C_m^n 可以定义阶乘函数，求出了阶乘，就可以求出 C_m^n。

根据分析，设计步骤如下：

(1) 设计界面如图 7-14 所示。

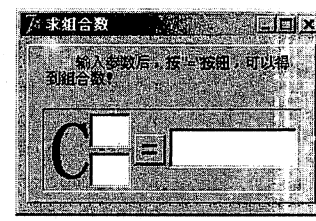


图 7-14 求组合数界面

(2) 编写 comb 组合函数，代码如下：

```
function comb(m,n:integer):real;

 var x,y,z:real;

 function fac(n:integer):real;

  var p:real;i:integer;

 begin
  p:=1;
  for i:=1 to n do p:=p * i;
   fac:=p;
 end;

begin
  x:=fac(m);
  y:=fac(n);
  z:=fac(m-n);
  comb:=x/(y * z);
end;
```

(3) 单击 Button1，编写 Button1 的 OnClick 事件过程。如下：

```
procedure TForm1.Button1Click(Sender: TObject);

 var m,n:integer;

begin
 n:=strtoint(edit1.Text);
 m:=strtoint(edit2.Text);
 if n>m
 then edit3.Text:='m 必须>=n!'
```

```
 else edit3.Text:=floattostr(comb(m,n));
end;
```

(4) 运行程序,如图 7-15 所示。

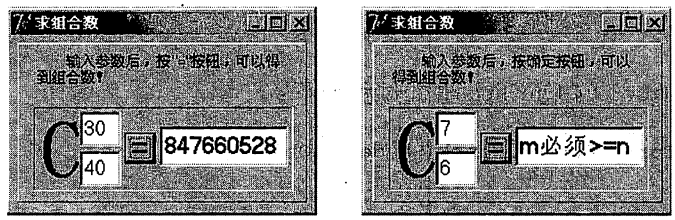


图 7-15　程序运行结果

说明：这里将 fac 函数和 comb 函数的返回值都定义为 real 类型。因为 integer 或者 int64 表示数的范围有限。如果定义为 int64 只能够求出 m 小于等于 20 以内的组合数；如果定义为 integer 则适用范围更小。

为了防止在编辑框中输入非法的字符,可以编写如下代码：

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
 if not((key>=#48)and(key<=#57)or(key=#8))   //可以输入数字和 BackSpace
 then key:=#0                                //#48 是字符"0",#57 是字符"9",#8 是 BackSpace
end;

procedure TForm1.Edit2KeyPress(Sender: TObject; var Key: Char);
begin
 if not((key>=#48)and(key<=#57)or(key=#8))
 then key:=#0
end;
```

【例 7-9】 输入 m、n 的值求组合数 C_m^n。要求使用嵌套过程的方法来做。

步骤如下：

(1) 设计界面,省略。

(2) 编写自定义过程,代码如下：

```
procedure comb(m,n:integer;var com:real);

 var x,y,z:real;

 procedure fac(n1:integer;var fa:real);

  var i:integer;

 begin
  fa:=1;
  for i:=1 to n1 do fa:=fa*i;
 end;
```

```
begin
 fac(m,x);
 fac(n,y);
 fac(m-n,z);
 com:=x/(y*z);
end;
```

(3) 编写 Button1 的 OnClick 事件过程如下：

```
procedure TForm1.Button1Click(Sender: TObject);

 var m,n:integer;c:real;

begin
 n:=strtoint(edit1.Text);
 m:=strtoint(edit2.Text);
 if n>m
 then edit3.Text:='m必须>=n!'
 else begin
       comb(m,n,c);
       edit3.Text:=floattostr(c);
      end;
end;
```

(4) 编写其他事件过程的代码如下：

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
 if not ((key>=#48) and (key<=#57) or (key=#8))
 then key:=#0
end;

procedure TForm1.Edit2KeyPress(Sender: TObject; var Key: Char);
begin
 if not ((key>=#48) and (key<=#57) or (key=#8))
 then key:=#0
end;
```

(5) 程序运行结果与例 7-9 完全一样。

7.4.2 子程序的递归

子程序直接或者间接地调用它自己就称之为子程序的递归，其可分为函数的递归和过程的递归。

例如：求自然数 n 的阶乘，可以使用递归定义为：

$$n! = \begin{cases} 1 & n = 0 \\ n \times (n-1)! & n > 0 \end{cases}$$

求 m 和 n 的最大公约数 gys，可以定义为：

$$\text{m 和 n 的公约数} = \begin{cases} m & n = 0 \\ \text{n 和 m 与 n 的余数的公约数} & n > 0 \end{cases}$$

从上面两个递归看出，要想定义递归子程序，必须符合两个条件：

(1) 一个规模比较大的问题，可以分解为若干个规模较小的同样类型的问题。

(2) 规模小到一定程度有固定的解。

1. 函数的递归

【例 7-10】 求自然数 n 的阶乘，已知：

$$n! = \begin{cases} 1 & n = 0 \\ n \times (n-1)! & n > 0 \end{cases}$$

分析：本题使用函数的递归，定义 fac(n)为 real 类型。

设计步骤为：

(1) 设计界面，如图 7-16 所示。

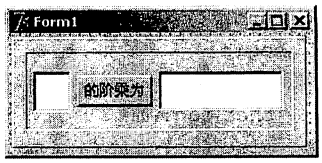


图 7-16　求阶乘的界面

(2) 编写自定义函数如下：

```
function fac(n:integer):real;
begin
 if n=0
 then fac:=1
 else fac:=n * fac(n-1);
end;
```

(3) 编写 Button1 的 OnClick 事件过程如下：

```
procedure TForm1.Button1Click(Sender: TObject);

 var n:integer;

begin
 n:=strtoint(edit1.Text);
 edit2.Text:=floattostr(fac(n))
end;
```

(4) 为了防止其他非法字符输入，可以编写过程：

```
procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
begin
 if not((key>=#48)and(key<=#57)or(key=#8))
 then key:=#0
end;
```

(5) 运行结果如图 7-17 所示。

【例 7-11】 求 m 和 n 的最大公约数。

分析：已知 m 和 n 的最大公约数等于 n 和 m mod n 的最大公约数，而 m 和 0 的最大公约数为 m。符合递归的两个条件。

步骤如下：

(1) 设计界面，如图 7-18 所示。

图 7-17 程序运行结果

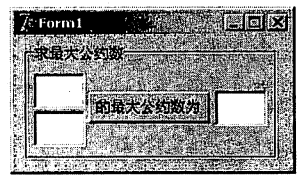


图 7-18 程序界面

(2) 编写递归的函数如下：

```
function fac(m,n:int64):int64;
begin
 if n=0
 then fac:=m
 else fac:=fac(n,m mod n)
end;
```

(3) 编写 Button1 的 OnClick 事件过程如下：

```
procedure TForm1.Button1Click(Sender: TObject);

 var a,b:int64;

begin
 a:=strtoint(edit1.Text);
 b:=strtoint(edit2.Text);
 edit3.Text:=inttostr(fac(a,b))
end;
```

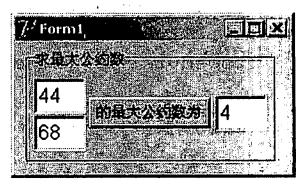


图 7-19 程序运行结果

(4) 运行结果如图 7-19 所示。

2. 过程的递归

【例 7-12】 n 阶汉诺(Hanno)塔问题。设有三个分别命名为 a、b、c 的塔座，在 a 上有 n 个直径大小不等的圆盘，从大到小放置(较小的圆盘放置在较大的圆盘上)。现要求将这 n 个圆盘移动到 c 座上，并且依旧按照原来的顺序放置(较小的圆盘放置在较大的圆盘上)可以使用 b 塔座过渡。圆盘移动时必须遵守如下规则：

① 每次只许移动一个盘子。

② 圆盘可以放置在任何一个圆盘上。

③ 任何时刻都不允许较大圆盘放置在较小圆盘上。

分析：这是典型的递归问题。实际上，将 n 个圆盘从 a 座移动到 c 座可以分 3 步进行。

① 将 a 座上面的 n－1 个较小的圆盘移动到 b 座，中间可以经过 c 座过渡。

② 再将 a 座上最大的那个圆盘直接移动放置在 c 盘上。

③ 再将 b 盘上的 n－1 个圆盘全部移动到 c 盘上，中间可以经过 a 过渡。

这 3 步都是规模较小的同类的问题，而且当只有一个圆盘时可以直接移动。

根据分析，设计步骤如下：

(1) 设计界面，如图 7-20 所示。

(2) 编写递归过程如下：

```
procedure hanno(a,b,c:char;n:integer);

 var s:string;

begin
 if n=1
 then begin
       s:='Move  '+a+'  to  '+c;          //n=1,只有一步可以完成移动
       form1.Memo1.Lines.Add(s);          //把移动方法输出在 memo 中
      end
 else begin                               //n>1,分为 3 步进行
       hanno(a,c,b,n-1);                  //第①步
       s:='Move  '+a+'  to  '+c;          //第②步
       form1.Memo1.Lines.Add(s);
       hanno(b,a,c,n-1);                  //第③步
      end;
end;
```

(3) 编写 Button1 的 OnClick 事件过程如下：

```
procedure TForm1.Button1Click(Sender: TObject);

 var first,second,last:string[1];n:integer;

begin
 memo1.Lines.Clear;                       //清空 memo
 first:=trim(edit1.Text);                 //初始塔座
 second:=trim(edit2.Text);                //过渡塔座
 last:=trim(edit3.Text);                  //目标塔座
 n:=strtoint(edit4.Text);                 //圆盘个数
 hanno(first[1],second[1],last[1],n)
end;
```

(4) 运行结果，如图 7-21 所示。

图 7-20　程序界面

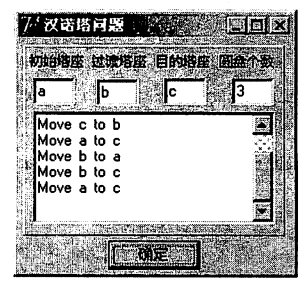


图 7-21　程序运行结果

问题：可否将 n 阶汉诺塔问题分解成下面 3 步进行？

① 将 a 座上面的最小的圆盘移动到 b 座。

② 再将 a 座上剩下的 n－1 的圆盘移动到 c 座，中间可以经过 b 过渡。

③ 再将 b 盘上的那个最小的圆盘移动到 c 盘上。

请读者思考。

7.5 变量的作用域

变量的作用域是指变量在哪些范围可以使用。当一个程序包含多个子程序的时候，各个子程序中都可以定义变量，这些变量只能够在所定义的子程序中有效，称之为局部变量，而在 Delphi 中还允许用户定义全局的变量。

7.5.1 公有变量和私有变量

在单元接口部分声明的变量称为公有变量，公有变量可以被其他单元引用。单元接口 interface 决定了本单元哪些变量是公有变量。其他单元如需使用该公有变量只需要在 uses 部分加上该单元即可。

在单元的实现部分(implementation)部分声明的变量属于死右边量，只能被本单元使用。

7.5.2 全局变量和局部变量

在子程序中定义的变量，只能够在该子程序中有效，称之为局部变量。而在子程序之前，在 Implementation 之后定义的变量可以被本单元所有子程序所使用，属于全局变量。

一般来说，要尽量多地使用局部变量，防止全局变量的副作用，如果需要在多个子程序之间交换数据则可以通过全局变量来进行。

【例 7-13】 在例 7-12 中，如果希望在移动圆盘之前显示第多少步，此时，可以考虑使用全局变量。

步骤省略，修改后的程序如下：

```
implementation
{$R* .dfm}

var count:int64;                    //定义全局变量,用于表示移动的次数

procedure hanno(a,b,c:char;n:integer);

 var s:string;

begin
 if n=1
 then begin
```

```
        count:=count+1;
          //无论哪次调用过程移动圆盘,count 值都有效,也就是在上次基础上再加 1
        s:='第'+inttostr(count)+'步,Move  '+a+'  to  '+c;
        form1.Memo1.Lines.Add(s);
       end
  else begin
        hanno(a,c,b,n-1);
        count:=count+1;
          //无论哪次调用过程移动圆盘,count 值都有效,也就是在上次基础上再加 1
        s:='第'+inttostr(count)+'步,Move  '+a+'  to  '+c;
        form1.Memo1.Lines.Add(s);
        hanno(b,a,c,n-1);
       end;
end;

procedure TForm1.Button1Click(Sender: TObject);

 var first,second,last:string[1];n:integer;

begin
 count:=0;                    //移动之前 count 先赋值为 0
 memo1.Lines.Clear;
 first:=trim(edit1.Text);
 second:=trim(edit2.Text);
 last:=trim(edit3.Text);
 n:=strtoint(edit4.Text);
 hanno(first[1],second[1],last[1],n)
end;
```

运行结果如图 7-22 所示。

【例 7-14】 编写一个计算器程序。

步骤如下：

(1) 添加加速按钮 SpeedButton1～SpeedButton16，添加编辑框 Edit1，添加 BitBtn 组件 BitBtn1 和 BitBtn2。

(2) 调整这些组件的大小和位置，设置属性，见表 7-1 所示。界面如图 7-23 所示。

图 7-22　汉诺塔运行结果

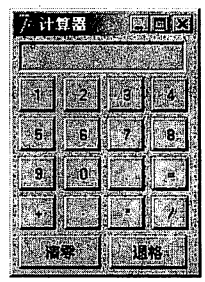


图 7-23　计算器界面

表 7-1 组件的属性设置

组　件	属　性	属　性　值
SpeedButton1～SpeedButton11	Caption	1、2、3～9、0
SpeedButton12～SpeedButton15	Caption	+、-、*、/
SpeedButton16	Caption	=
BitBtn1	Caption	清零
BitBtn2	Caption	退格

(3) 首先定义全局变量：

```
implementation
{$R *.dfm}

var op:char;one,two,result:real;        //定义全局变量
```

(4) 因为 SpeedButton1～SpeedButton9 的代码类似，故同时选中数字按钮 1～9，即 SpeedButton1～SpeedButton9。在 Events 页中选定 OnClick 事件，双击右边空白栏，编写事件过程 TForm1. SpeedButton1Click。这样 SpeedButton1～SpeedButton9 就共享了该事件过程。代码如下：

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
 edit1.Text:=edit1.Text+(sender as Tspeedbutton).caption;
  //所按键的 Caption 显示在编辑框
 if op=#0                                  //尚未输入运算符,即目前所按数字键是第 1 个操作数
 then begin
       one:=strtofloat(edit1.text);        //第 1 个操作数
       edit1.Text:=floattostr(one)         //显示在编辑框中
      end
 else begin                                //如果 op<>#0,表示是第 2 个操作数
       two:=strtofloat(edit1.text);        //第 2 个操作数
       edit1.Text:=floattostr(two)         //显示第 2 个操作数在编辑框
      end;
end;
```

编写“0”的事件过程如下：

```
procedure TForm1.SpeedButton10Click(Sender: TObject);
begin
 if not ((pos('.',edit1.Text)=0)and (pos('0',edit1.Text)=1))
  //如果编辑框中无小数点,且第一个字符是 0,则不许继续加入 0 字符
 then edit1.Text:=edit1.Text+SpeedButton10.Caption;    //允许加入 0 字符
  //将编辑框中的数字保存到操作数 1 或者操作数 2
 if op=#0                                  //尚未输入操作运算符号
 then one:=strtofloat(edit1.text)          //保留到第 1 个操作数中
 else two:=strtofloat(edit1.text)          //保留到第 2 个操作数中
```

```
end;
```

编写小数点 SpeedButton11 的事件过程，如下：

```
procedure TForm1.SpeedButton11Click(Sender: TObject);
begin
 if pos('.',edit1.Text )=0                    //无小数点则可以输入小数点
 then edit1.Text:=edit1.Text +SpeedButton11.Caption;
 if op=#0                                     //如果是第 1 个操作数
 then one:=strtofloat(edit1.text)
 else two:=strtofloat(edit1.text)             //第 2 个操作数
end;
```

选择“+”、“-”、“*”和“/”，即 SpeedButton12～SpeedButton15，编写共享的事件过程。如下：

```
procedure TForm1.SpeedButton12Click(Sender: TObject);

 var s:string;

begin
 case op of
   //在按当前运算符号之前，如果事先已经按过运算符，则先计算前面的运算
   //并把结果作为当前运算的第 1 操作数
  '+':result:=one+two;
  '-':result:=one-two;
  '*':result:=one*two;
  '/':result:=one/two;
  #0:result:=one;                             //本次运算属于第 1 次运算
 end;
 edit1.Text:=floattostr(result);
 one:=result;
 s:=(sender as Tspeedbutton).caption;
 op:=s[1];
 edit1.Text:='';
end;
```

编写“=”号，即 SpeedButton16 的 OnClick 事件过程。如下：

```
procedure TForm1.SpeedButton16Click(Sender: TObject);
begin
 case op of
  '+':result:=one+two;
  '-':result:=one-two;
  '*':result:=one*two;
  '/':result:=one/two;
  #0:result:=one;
 end;
 edit1.Text:=floattostr(result);
  //按“=”之后，可以把编辑框中的结果作为下次运算的第 1 个操作数
 one:=result;
```

```
 op:=#0;                                    //下次运算的操作符号还未确定
end;
```

按钮“清零”,即 BitBtn1 的 OnClick 事件过程如下:

```
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
 edit1.Text:='';
 one:=0;
 two:=0;
 op:=#0;
end;
```

按钮“退格”,即 BitBtn2 的 OnClick 事件过程如下:

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
 edit1.Text:=copy(edit1.Text,1,length(edit1.Text)-1) ;
 if(op=#0)and (length(edit1.text)>0)
 then one:=strtofloat(edit1.text);
 if (op=# 0)and (length(edit1.text)=0)
 then one:=0 ;
 if(op<>#0)and(length(edit1.text)>0)
 then two:=strtofloat(edit1.text);
 if (op<>#0)and (length(edit1.text)=0)
 then two:=0 ;
end;
```

(5) 运行程序:

```
① 34+5+6-7=38*6-2=226                  //波浪线为程序显示结果
② 2-6*3=-12+.6=-11.4                   //波浪线为程序显示结果
```

7.6 小结

本章讲述了过程、函数、参数传递、子程序的嵌套与递归和变量的作用域。其中函数与过程是本章的重点,递归是本章的难点;灵活使用递归可以使复杂的问题简单化,甚至迎刃而解。通过本章的学习,读者应该学会定义函数和过程,并学会使用自定义的函数和过程;还要掌握参数传递的规则;理解和掌握递归;学会使用共有变量和私有变量,知道在什么情况下使用私有变量,什么情况下使用共有变量。

习题

1. 定义一个 function,该 function 可以产生一个随机的小写英文字母。
2. 定义一个 function,该 function 可以产生一个随机的英文字母。
3. 定义一个 procedure,要求使用变参,用于判断一个数是否为素数。

4. 定义一个函数，用于判断某个数是否是 7 的倍数且不是 3 的倍数。并将 1000 以内的这样的数求和。

5. 已知一个自定义函数定义如下：

```
function fu(x,y,z:integer):integer;
begin
fu:=z-y div x;
end;
```

则表达式 fu(2,4,fu(6,8,12))的值为多少？

6. 定义一个递归函数，求 1＋2＋3＋…＋n。

7. 定义一个递归过程，求 1＋(1＋2)＋(1＋2＋3)＋…＋(1＋2＋3＋…＋n)。

8. 已知自定义函数声明如下：

```
procedure ab(a:integer;var b:integer);
begin
 a:=a+1;
 b:=b+10;
end;
```

执行下面的程序段：

```
x:=1;y:=2;ab(x+1,y);
```

则 x 和 y 值分别是多少？

9. 定义一个递归过程，打印如图 7-24 所示的图形。

10. 验证“哥德巴赫”猜想。界面如图 7-25 所示。将任何一个大于 4 的偶数分解成两个素数相加的形式。要求定义一个函数，用于判断一个数是否是素数。另外，再定义一个过程，用于将一个较大的偶数分解成两个素数相加的形式，并把结果添加到多行编辑框中，如图 7-26 所示。

图 7-24　打印图形

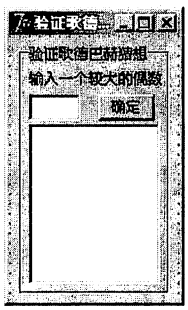


图 7-25　程序界面

图 7-26　程序运行结果

11. 使用递归和非递归的方法编写函数，求：

$$f(x,n) = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!}$$

12. 编写过程打印 n 行杨辉三角。如图 7-27 所示。

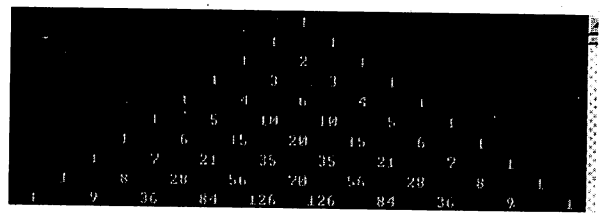


图 7-27 杨辉三角

13. 已知下列数列：

$$f(n)=\begin{cases}1 & n=1,2\\ f(n-1)+f(n-2) & n>2\end{cases}$$

请编写递归函数求解。

第 8 章

对 话 框

对话框是人机对话的最重要形式，使得用户与程序交互极为方便。Delphi 提供了标准的对话框函数(或过程)和对话框组件。

对话框函数(或过程)有：ShowMessage、ShowMessageFmt、MessageDlg、InputBox、InputQuery。

对话框组件在 Dialog 组件面板中，如图 8-1 所示。有 OpenDialog、SaveDialog、OpenPictureDialog、SavePictureDialog、FontDialog、ColorDialog、PrintDialog、PrintSetdialog、FindDialog 和 ReplaceDialog 等。

图 8-1 对话框组件

8.1 对话框函数(或过程)

我们可以将对话框函数(或过程)分成输出类函数(或过程)和输入类函数(或过程)两类。ShowMessage、ShowMessageFmt 和 MessageDlg 这 3 个函数是输出类(消息类)过程，主要是将结果输出。用户根据程序的需要可以选择合适的输出过程。而 InputBox 和 InputQuery 这两个函数主要是向程序输入数据，用户也可以选择适合自己程序的输入函数。

8.1.1 输出类对话框过程

1. ShowMessage 过程

ShowMessage 函数用来显示一个简单的信息，语法格式为：

```
ShowMessage(<信息内容>);
```

例如语句：

```
showmessage('欢迎使用本系统!');
```

显示的对话框如图 8-2 所示。

2. ShowMessageFmt 过程

ShowMessageFmt 过程用来显示一个简单的信息，该信息以一定的格式来显示语法。

```
ShowMessageFmt(<信息内容>,<参数组>);
```

例如语句：

```
showmessageFmt('%s 今年%d岁。',['Tom',27]);
```

显示的对话框如图 8-3 所示。

图 8-2 ShowMessage 对话框

图 8-3 ShowMessageFmt 对话框

说明：本处的信息内容是格式化了的字符串，该字符串需要和参数组配合使用，格式字符串和参数组的使用方法参见前面讲的 format 函数。

3. MessageDlg 函数

MessageDlg 函数用来显示一个信息对话框，并等待用户响应，根据用户的响应对话框返回一个值。程序可以根据返回值的不同来执行不同的操作，因此该函数的交互能力比前面讲的两个过程强。

调用 MessageDlg 函数在屏幕中央显示信息对话框，其语法格式为：

```
<变量>:=MessageDlg(<信息内容>,<类型>,<按钮组合>,HelpCtx);
```

说明：

(1) 变量是 Word 类型，其值与按下的按钮有关。

(2) ＜信息内容＞是显示在对话框中的信息。

(3) ＜类型＞是对话框的类型，即对话框显示的图标，其意义参见表 8-1。

表 8-1 对话框类型

取　值	说　明
mtWarning	含有黄色感叹号的警告对话框
mtError	含有红色叉符号的错误对话框
mtInformation	含有蓝色 i 符号的信息对话框
mtConformation	含有蓝色问号的确认对话框
mtCustom	不含图标的一般对话框

(4) ＜按钮组合＞是对话框中出现按钮的组合形式，参见表 8-2。

表 8-2　按钮组合中的按钮

取　值	说　明
mbYes	Yes 按钮，按此键返回 mrYes 或者 6
mbNo	No 按钮，按此键返回 mrNo 或者 7
mbOK	OK 按钮，按此键返回 mrOK 或者 1
mbCancel	Cancel 按钮，按此键返回 mrCancel 或者 2
mbHelp	Help 按钮
mbAbort	Abort 按钮，按此键返回 mrAbort 或者 3
mbRetry	Retry 按钮，按此键返回 mrRetry 或者 4
mbIgnore	Ignore 按钮，按此键返回 mrIgnore 或者 5
mbAll	All 按钮，按此键返回 mrAll 或者 8
mbNoToAll	NoToAll 按钮，按此键返回 mrNoToAll 或者 9
mbYesToAll	YesToAll 按钮，按此键返回 mrYesToAll 或者 10

按钮组合是集合的形式，如[mbYes，mbNo，mbCancel]表示对话框有 3 个按钮，分别是 Yes 按钮、No 按钮和 Cancel 按钮。

(5) HelpCtx 是当用户单击 Help 按钮或者按 F1 键时，显示的帮助主题。

例如：下面是对话框函数的举例，如图 8-4 所示。

```
var a:word;
...
a:=messagedlg('文件已修改,是否保存?',mtConfirmation,
[mbyes,mbNo],1);
if a=mryes then              //在此保存文件
```

图 8-4　对话框举例

8.1.2　输入类对话框函数

1. InputBox 函数

InputBox 函数显示一个供用户输入的对话框，并返回用户输入的信息。其语法格式为：

```
<变量>:=InputBox(<对话框标题>,<信息内容>,<默认内容>);
```

说明：

(1) ＜变量＞是字符串变量，是函数返回值，即输入的字符串。

(2) ＜对话框标题＞是指对话框的标题，是字符串类型。

(3) ＜信息内容＞是指对话框中出现的提示文本，可以在信息内容中加入回车符(#13)，这样在信息框中可以显示多行的提示文本。

(4) ＜默认内容＞指定对话框的输入框中显示的默认文本，可以修改。如果用户单击“确定”按钮，则返回输入框中的文本；如果用户按“取消”按钮，则返回默认内容到

变量。

例如：下面是 InputBox 函数的举例。

```
var xm:string;age:integer;
...
xm:=inputbox('输入','输入姓名','张三');
age:=strtoint(inputbox('输入','输入年龄','22'));
showmessagefmt('%s 同志今年%d岁。',[xm,age]);
```

运行结果如图 8-5 所示。

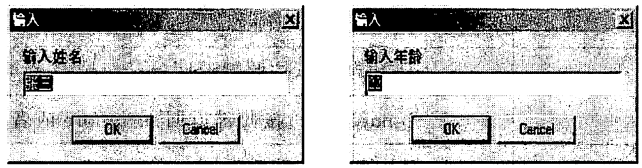


图 8-5 输入对话框

说明：输入的年龄是整型，所以需要将字符类型转化为整型。

2. InputQuery 函数

InputQuery 函数与 InputBox 函数类似，但是它的功能更为强大(可对按 OK 和按 Cancel 进行处理)，它通过字符串变量得到输入的值，而函数本身返回的却是 Boolean 型的值。也就是说按 OK 按钮就返回 True，按 Cancel 按钮返回 False。

InputQuery 函数的格式：

```
<变量>:=InputQuery(<对话框标题>,<信息内容>,<字符串变量>);
```

说明：

(1) <变量>是 Boolean 型变量。按“确定”返回 True，按“取消”返回 False。

(2) <对话框标题>是指对话框的标题，是字符串类型。

(3) <信息内容>是指对话框中出现的提示文本，可以在信息内容中加入回车符(#13)，这样在信息框中可以显示多行文本。

(4) <字符串变量>是事先定义好的字符串变量。如果按 OK 按钮则将输入的字符串保存到字符串变量，同时函数返回 True；如果按 Cancel 按钮则字符串变量的值不变，函数返回 False。

下例就是 InputQuery 函数的举例。

```
var xm:string;
...
xm:='张三';                                    //给姓名赋初值
if inputQuery('输入','输入姓名',xm)
then{按 OK 按钮,姓名是输入框中输入的字符串}
else{按 Cancel 按钮,姓名还是张三};
showmessage('我叫'+xm);
```

说明：最好给 xm 赋一个初始值，否则，最后得到的 xm 可能是空字符。

8.2　对话框组件

Delphi 提供了很多对话框组件。通过这些组件，用户可以打开、保存文件，选择字体、颜色，打印设置和打印，查找和替换。需要使用对话框时将相应的对话框放置在窗体上，设置属性值，以 Execute 方法运行对话框，通过返回值获得对话框的运行情况及相关属性值。

8.2.1　文件类对话框组件

文件类对话框组件主要有 OpenDialog、SaveDialog、OpenPictureDialog 和 SavePictureDialog 4 个，它们可以对文件进行打开和保存操作。

1. OpenDialog 打开对话框

执行 OpenDialog 组件的 Execute 方法，将显示“打开”对话框，如图 8-6 所示。当用户单击对话框的“打开”按钮时，Execute 返回 True，单击“取消”按钮时，返回 False。

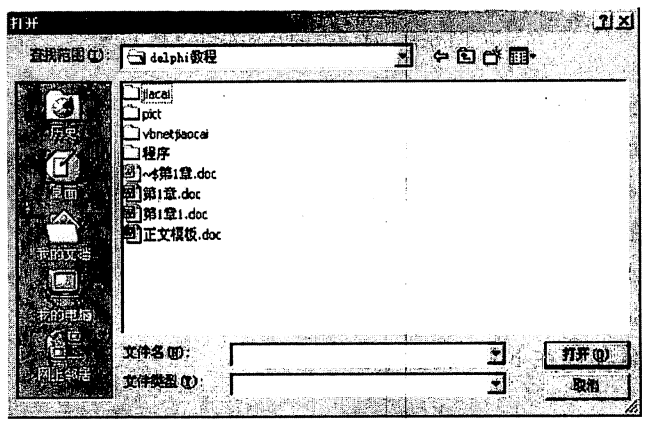


图 8-6　“打开”对话框

OpenDialog 的常用属性如下。

(1) Title 属性：对话框的标题。

(2) InitialDir 属性：初始显示的文件目录。

(3) FileName 属性：选中文件的文件名(包含路径在内)。

(4) Options 属性：指定对话框的参数，含多个参数，将在下面介绍少数几个。

(5) Filter 属性：文件过滤器。

过滤器的用法举例：

```
opendialog1.Filter:='Word 文档|*.doc|纯文本文件|*.txt|所有文件(*.*)|*.*';
```

该语句设置了过滤器，从该过滤器看，当前打开对话框可以有三种过滤方式。第一种是 Word 文档，过滤器设置为“Word 文档|*.doc”，其中扩展名是.doc，前面的“Word 文档”是显示提示信息；第二种是纯文本文件，过滤器设置为“纯文本文件|*.txt”，其中扩

展名是 txt，提示文字是“纯文本文件”；第三种过滤器是“所有文件(*.*)|*.*”，前面“所有文件(*.*)”是提示文字，后面的“*.*”可以在对话框中显示所有文件。如图 8-7 所示。

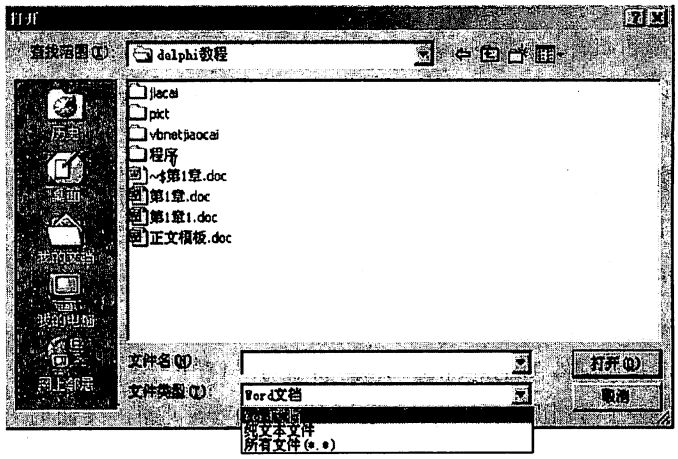


图 8-7 对话框的过滤器设置

上述过滤器是用代码来实现的，也可以在属性中设置，单击 Filter 属性后的[...]，出现过滤器编辑框，设置如图 8-8 所示过滤器。

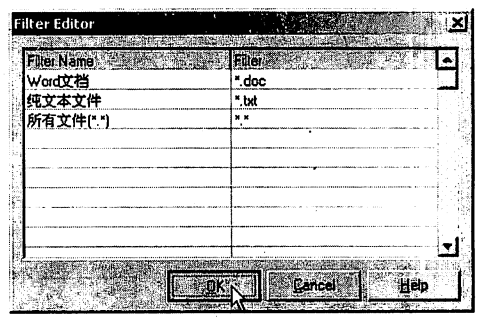


图 8-8 使用过滤器编辑器设置过滤器

(6) FilterIndex 属性：文件过滤器的索引，所选过滤器项的序号，序号从 1 开始。

(7) ofAllowMultiSelect 属性：在 Options 中，ofAllowMultiSelect 值为 True 时，可以选择多个文件，文件名存放在 Files 属性中。

例如：

```
if opendialog1.Execute then
begin
 memo1.Lines:= opendialog1.Files ;
 label1.Caption:= opendialog1.Files[0]
end;
```

说明：memo1. Lines :=opendialog1. Files 语句表示将选定的所有文件的文件名(含路径)显示在 Memo1 多行编辑框中，而 label1. Caption :=opendialog1. Files[0]表示将选定的第一个文件的文件名显示在标签 Label1 中。选定的文件个数为：OpenDialog1. files. count。

OpenDialog 的主要事件：

(1) OnFolderChange 事件：当文件目录更改时触发该事件。

(2) OnSelectionChange 事件：选择的文件发生变化时触发该事件。

(3) OnTypeChange 事件：文件类型发生改变时触发该事件。

2. SaveDialog 保存对话框

执行 SaveDialog 组件的 Execute 方法，将显示“另存为”对话框，如图 8-9 所示。当用户单击对话框的“打开”按钮时，SaveDialog 组件不能够选择多个文件。它的很多属性与 OpenDialog 一样，特殊属性如 DefaultExt 属性(保存文件的默认扩展名)。

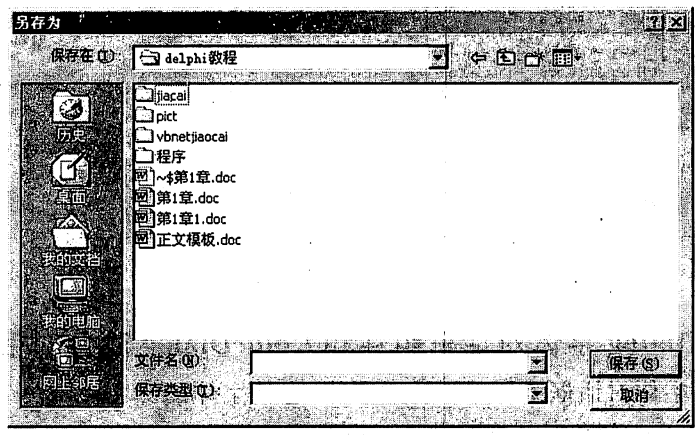


图 8-9　“另存为”对话框

例如：

```
savedialog1.DefaultExt:='txt';          //扩展名是 txt
if savedialog1.Execute then
memo1.Lines.SaveToFile(savedialog1.FileName);
```

说明：默认扩展名无须加“.”号。

3. OpenPictureDialog 和 SavePictureDialog 对话框

这两个对话框和 OpenDialog 对话框、SaveDiaolg 对话框类似，操作方法也类似，不同的是 OpenPictureDialog 和 SavePictureDialog 这两个对话框右边多了一个用于浏览图片的区域，如图 8-10 和图 8-11 所示。

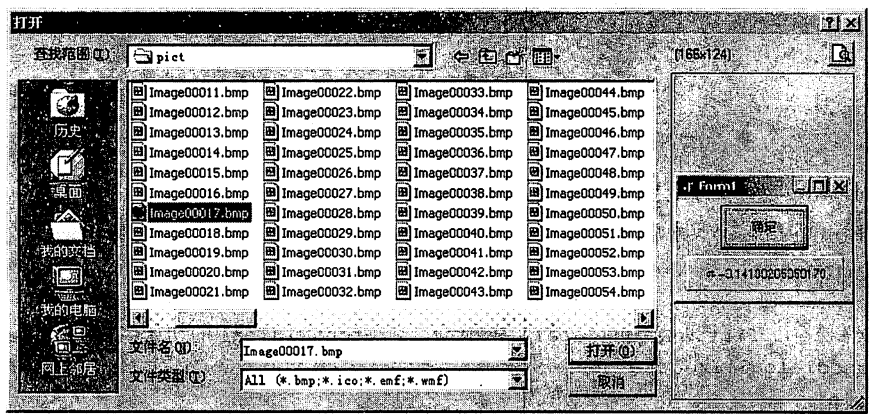


图 8-10　OpenPictureDialog 对话框

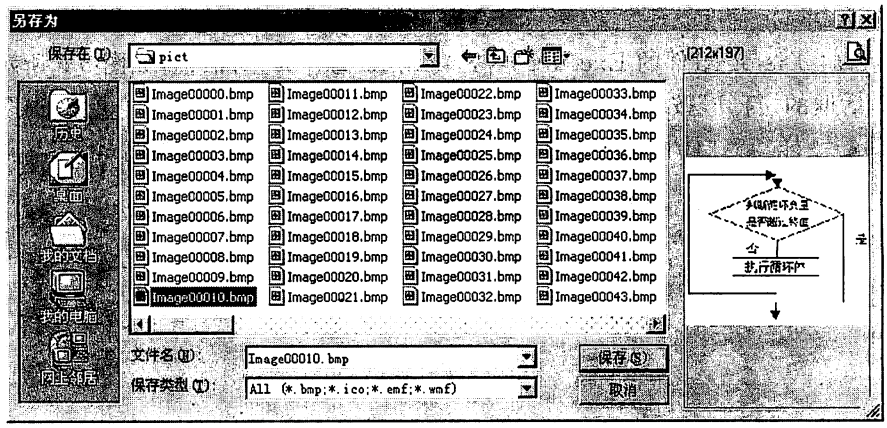


图 8-11 SavePictureDialog 对话框

8.2.2 FontDialog 字体对话框组件和 ColorDialog 颜色对话框组件

FontDialog 字体对话框组件用于设置字体，包括字体的颜色、字体名称、大小、形状、效果等。ColorDialog 颜色对话框组件用于设置颜色。

1. FontDialog 字体对话框组件

FontDialog 对话框用于设置字体，执行 FontDialog 的 Execute 方法可以显示字体对话框，如图 8-12 所示。主要属性是 Font 属性。

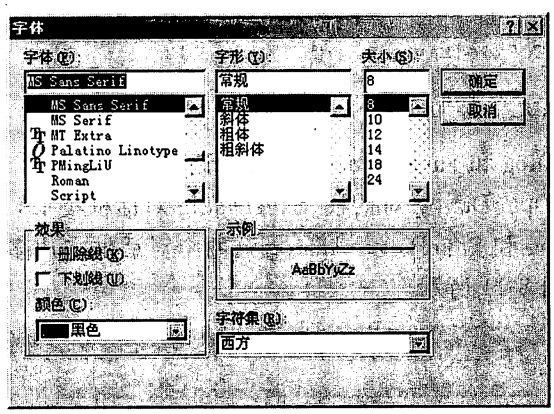


图 8-12 FontDialog 字体对话框

FontDialog 组件的主要属性如下。

(1) Font 属性：字体属性。

用法如下：

```
fontdialog1.Font.Assign(Label1.Font);    //fontdialog1 的初始字体与 Label1 的一样
if fontdialog1.Execute then
  begin
   label1.Font.Size:=fontdialog1.Font.Size;
   label1.Font.Name:=fontdialog1.Font.Name;
```

```
    label2.Font.Assign(fontdialog1.Font);
  end;
```

说明：Label1 的字体和大小分别取自 FontDialog1，而 Label2 的字体、字号、颜色、形状等一次性取自 FontDialog1，这就是 FontDialog 的不同用法。而第一个语句的作用是“字体”对话框的初始字体与 Label1 一致。

(2) fdEffects 属性：在 Options 中，用于设置字体的形状、颜色等。默认值为 True 表示显示字体的效果、颜色等。图 8-13 所示为其值是 False 时的“字体”对话框。

2. ColorDialog 颜色对话框组件

ColorDialog 颜色对话框组件用于设置颜色。执行组件的 Execute 方法，可以显示“颜色”对话框，如图 8-14 所示。

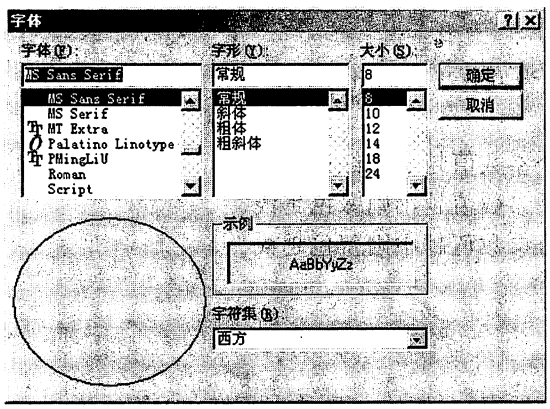


图 8-13　fdEffects 值为 False 时的“字体”对话框（注意左下角与图 8-12 的区别）

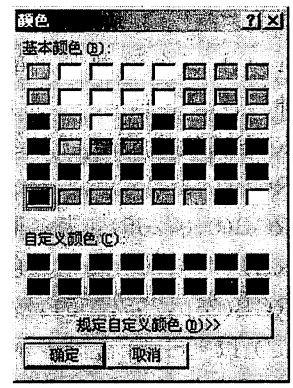


图 8-14　“颜色”对话框

ColorDialog 对话框的主要属性如下。

(1) Color 属性：颜色。

例如：

```
if colordialog1.Execute
then label1.Color:=colordialog1.Color
```

设置标签 Label1 的颜色为“颜色”对话框中选定的颜色。

(2) cdFullOpen 属性：是否全部展开“颜色”对话框，默认值为 False。当值为 True 时，展开自定义颜色部分，如图 8-15 所示。

3. 关于颜色

颜色在 Delphi 中是 Tcolor 类，在 Graphics 单元中 Tcolor 定义为一个 32 位的二进制整数。在 Delphi 中表示颜色的方法如下。

(1) 以 RGB 函数表示颜色。

由于颜色是由红、绿、蓝三个基本颜色合成的，如果红、绿、蓝所占比例不一样，表示的颜色当然也不一样。RGB 函数格式为：

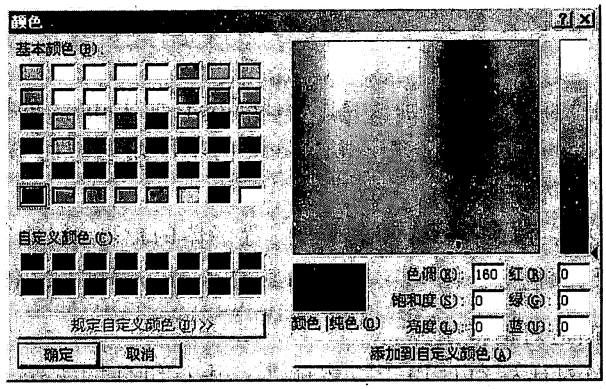


图 8-15 展开自定义颜色部分的“颜色”对话框

```
RGB(r,g,b)
```

其中 r,g,b 三个整型变量的取值范围为 0～255,因此 RGB 函数可以表示 256×256×256 种颜色。下面是一些常用的颜色:

RGB(255,0,0):表示红色,第一个参数值越小则红色越浅。

RGB(0,255,0):表示绿色,第二个参数值越小则绿色越浅。

RGB(0,0,255):表示蓝色,第三个参数值越小则蓝色越浅。

RGB(0,0,0):表示黑色。

RGB(255,255,255):表示白色。

(2) 以 Delphi 常量表示颜色。

表示颜色的常量前面冠以 cl(color 缩写)。例如,clRed 表示红色,clGreen 表示绿色,clBlue 表示蓝色,clBlack 表示黑色,clWhite 表示白色等。

8.2.3 FindDialog 查找对话框组件和 ReplaceDialog 替换对话框组件

在文件编辑的过程中经常用到查找和替换操作。FindDialog 对话框和 ReplaceDialog 对话框可以实现查找与替换功能。这两个对话框比前面所讲的对话框要复杂,其交互能力也比较强。

1. FindDialog 查找对话框组件

执 FindDialog 对话框的 Execute 方法可以显示“查找”对话框,如图 8-16 所示。

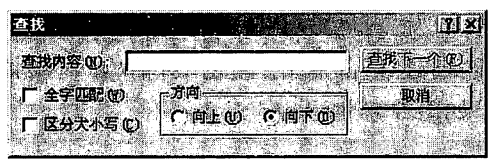


图 8-16 “查找”对话框

FindDialog 对话框的主要属性如下。

(1) FindText 属性:用户输入的查找内容字符串。

(2) Options 属性:用于设置查找规则,例如全字符匹配、大小写区分、查找方向等。

FindDialog 对话框的主要事件如下。

(1) OnFind 事件：单击“查找下一个”时触发该事件。

(2) OnClose 事件：单击“取消”时触发该事件。

2. FindDialog 查找对话框组件的应用

【例 8-1】 在 RichEdit 中查找字符串，找到该字符串则选定它，找不到则显示“查找结束”信息。

分析：此处要使用 FindDialog 对话框，另外还要使用到字符串函数，找到字符串还要选定该字符串，因此会使用到 RichEdit 组件，该组件提供了 SelStart 和 SelLength 方法，使用这两个方法可以实现字符的选定。根据分析，可知程序要用到 RichEdit、FindDialog 等组件。

步骤如下：

(1) 添加组件 RichEdit、FindDialog、Button，分别命名为 RichEdit1、FindDialog1 和 Button1，并调整这三个组件的位置和大小，如图 8-17 所示。

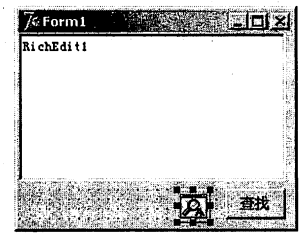


图 8-17　查找字符串程序界面

(2) 编写程序如下：

单击 Button1 显示“查找”对话框，TForm1. Button1Click 过程如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 finddialog1.Execute
end;
```

单击“查找下一个”，触发 OnFind 事件，OnFind 事件过程如下：

```
procedure TForm1.FindDialog1Find(Sender: TObject);

 var s,findtext:string;n,i,k:integer;

begin
 findtext:=finddialog1.FindText;
 n:=length(findtext);
 i:=richedit1.SelStart+richedit1.SelLength;                    //查找字符的起始位置
 s:=copy(richedit1.Text,i+1,length(richedit1.Text)-i);         //在 s 字符串中查找
 k:=pos(findtext,s);
 if k>0 then
  begin
   form1.Show;
   richedit1.HideSelection:=false;
   richedit1.SelStart:=i+k-1;
   richedit1.SelLength:=n;
  end
 else showmessage('搜索完毕!');
end;
```

说明：

(1) 关于 pos()函数，它用于判断某个字符串在另外一个字符串中的位置。例如，执

行“i :=pos('123','12345');”后 i=1,表示前面一个字符串在后面一个字符串中的位置(位置从 1 开始)。而执行“j :=pos('123','12345');”后 j=0,表示前面一个字符串不是后面字符串的子串。

(2) HideSelection 属性设置为 False 用于将选定的文字以反底形式显示。

(3) 设置 SelStart 属性和 SelLength 属性用于选定文字。

3. ReplaceDialog 替换对话框组件

执行 ReplaceDialog 对话框的 Execute 方法将显示“替换”对话框,如图 8-18 所示。

ReplaceDialog 对话框的主要属性如下。

(1) FindText 属性:用户输入的查找内容字符串。

(2) ReplaceText 属性:用户输入的用于替换的字符串。

(3) Options 属性:用于设置查找规则,例如,全字符匹配、大小写区分等。

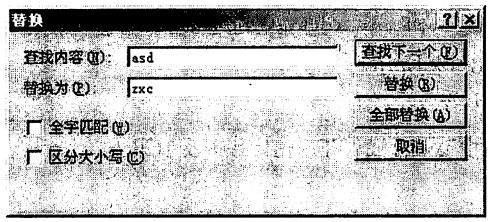


图 8-18 “替换”对话框

ReplaceDialog 对话框的主要事件如下。

(1) OnFind 事件:单击“查找下一个”按钮时触发该事件。

(2) OnClose 事件:单击“取消”按钮时触发该事件。

(3) OnReplace 事件:单击“替换”或者“全部替换”时触发该事件。

当单击“替换”按钮时,Options 属性集合中有 frReplace 值;当单击“替换所有”按钮时,Options 属性集合中有 frReplaceAll 值。

4. ReplaceDialog 替换对话框组件的应用

【例 8-2】 在 RichEdit 中查找字符串,找到该字符串则选定它,找不到字符串则显示“查找结束”信息。如果找到字符串则替换它。

分析:此处要使用查找和替换,结合例 8-1 的分析可知程序要用到 RichEdit、ReplaceDialog 等组件。

步骤如下:

(1) 添加组件 ReplaceDialog、RichEdit、Button,分别命名为 ReplaceDialog1、RichEdit1 和 Button1,并调整好组件的大小和位置。

(2) 编写代码如下:

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 Replacedialog1.Execute
end;

procedure TForm1.ReplaceDialog1Find(Sender: TObject);
```

```
var s,findtext:string;n,i,k:integer;

begin
 findtext:=replacedialog1.FindText;
 n:=length(findtext);
 i:=richedit1.SelStart+richedit1.SelLength;                    //查找字符的起始位置
 s:=copy(richedit1.Text,i+1,length(richedit1.Text)-i);         //在 s 字符串中查找
 k:=pos(findtext,s);
 if k>0 then
  begin
   form1.Show;
   richedit1.HideSelection:=false;
   richedit1.SelStart:=i+k-1;
   richedit1.SelLength:=n;
  end
 else showmessage('搜索完毕!');
end;

procedure TForm1.ReplaceDialog1Replace(Sender: TObject);
begin
if frReplace in ReplaceDialog1.Options then          //if...then...语句也可以省略
richedit1.SelText:=replacedialog1.ReplaceText;
end;
```

说明：选定的字符就是查找到的字符，语句“richedit1. SelText ：＝replacedialog1. ReplaceText;”的作用就是将选定的字符用“替换”对话框中的替换字符代替。请读者思考一下“替换所有”按钮的代码如何编写。

8.2.4　PrintDialog、PrinterSetupDialog 和 PageSetupDialog 对话框组件

这三个对话框组件与页面设置、打印设置和打印有关，其中 PageSetupDialog“页面设置”对话框组件是 7.0 版本以后才有的。

1. PrintDialog“打印”对话框组件

PrintDialog“打印”对话框用于设置与打印有关的一些属性，例如，选择打印机、打印范围、打印份数等。执行 PrintDialog 的 Execute 方法可以显示“打印”对话框，如图 8-19 所示。

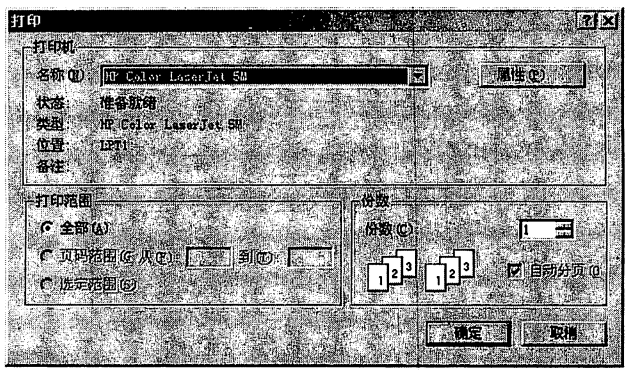


图 8-19　“打印”对话框

PrintDialog 对话框组件的属性如下。

(1) PrintRange 属性：打印范围，取值为 prAllPages 表示打印全部页面，取值为 prPageNums 表示页码范围数，值为 prPageSelection 表示选定打印范围。

(2) FromPage 与 ToPage 属性：PrintRange 属性值设置为 prPageNums 时有效，此时 FromPage 与 ToPage 表示打印的范围

(3) Copies 属性：表示打印份数。

2. PrinterSetupDialog“打印设置”对话框组件

PrinterSetupDialog 对话框用于设置纸张大小、打印方向、纸张来源等。执行 PrinterSetupDialog 的 Execute 方法可以显示“打印设置”对话框，如图 8-20 所示。

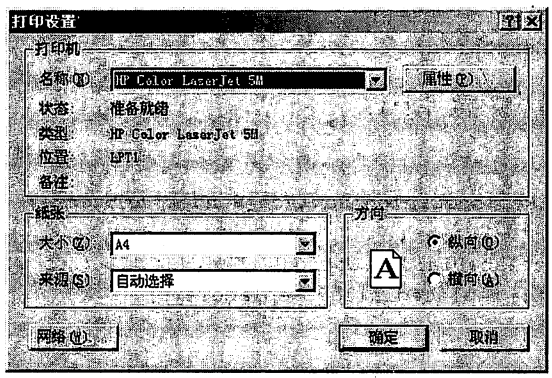


图 8-20 “打印设置”对话框

3. PageSetupDialog“页面设置”对话框组件

PageSetupDialog“页面设置”对话框用于设置纸张大小、打印方向、页边距等。执行 PageSetupDialog 对话框的 Execute 方法显示“页面设置”对话框，如图 8-21 所示。

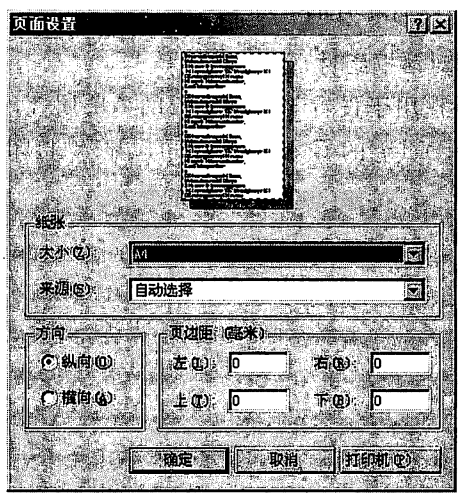


图 8-21 “页面设置”对话框

8.3　小结

本章不仅讲述了对话框函数(或过程),还讲述了系统提供的对话框组件。输入输出类函数(或过程)是程序中使用非常频繁的函数(或过程),读者务必熟练掌握。对话框组件是本章的难点且非常重要,特别是文件类对话框组件、字体对话框组件和颜色对话框组件。通过本章的学习,读者一定要做到:

- 掌握常见的输入、输出标准函数和过程。
- 掌握文件类对话框组件的使用方法。
- 掌握 FontDialog 和 ColorDialog 组件。
- 会使用 PrintDialog、PrinterSetupDialog 和 PageSetupDialog 组件。

习题

1. 窗体上有一个 Image 图片组件,利用 OpenPictureDialog 组件打开一个图片文件,将此图片文件显示在 Image 中。

2. 制作一个类似记事本的文本处理程序。

3. 使用 InputBox 或者 InputQuery 输入职工的个人信息,个人信息包括姓名、性别、家庭住址、年龄,并把这些信息显示在窗体的标签上。

4. 编写一个简单的编辑器,该编辑器可以打开和保存文件。界面如图 8-22 所示。

5. 在第 4 题中添加新建按钮,要求在新建文件之前判断该文件是否改变了。如果文件改变了则提示类似“文件被修改,是否保存?”的信息,如图 8-23 所示。如果单击 Yes 按钮则先保存再新建,如果单击 No 按钮,则不保存文件直接新建。

6. 在窗体上添加 RichEdit 组件 RichEdit1 和 FontDialog 组件 FontDialog1,添加 Button 组件 Button1(Caption 为“字体”)。单击“字体”按钮,为选定的文本设置字体。界面如图 8-24 所示。

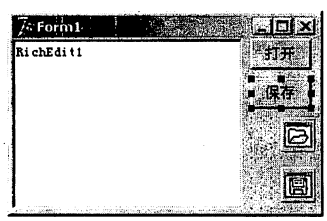


图 8-22　简单的编辑器

图 8-23　提示是否保存信息框

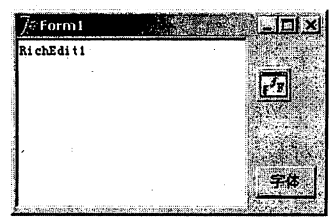


图 8-24　字体设置程序界面

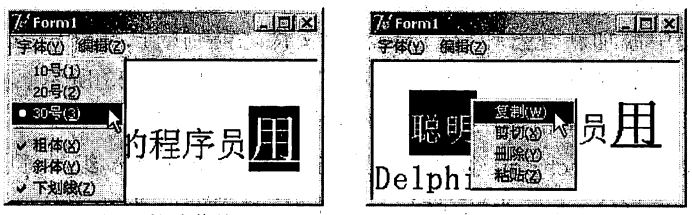
第 9 章

菜单、工具栏和状态栏

在 Windows 程序设计中，菜单、工具栏和状态栏是应用程序重要的界面元素。菜单、工具栏和状态栏可以使程序操作更加方便，界面更加美观、友好。

9.1 菜单

在 Delphi 中菜单分为下拉式菜单和弹出式菜单，使用 MainMenu 组件可以很方便地制作出下拉式菜单，使用 PopUpMenu 组件也可以方便地制作出弹出式菜单。下拉式菜单单击主菜单项即可显示出来，而弹出式菜单则可以通过鼠标右键显示出来，如图 9-1 下标。



(a) 下拉式菜单　　(b) 弹出式菜单

图 9-1　下拉式菜单和弹出式菜单

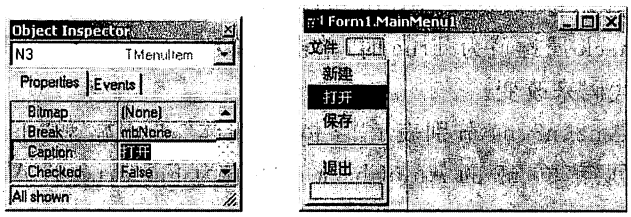
9.1.1 下拉式菜单组件 MainMenu

MainMenu 下拉式菜单中菜单项保存在 Items 属性中，Items 属性是 MainMenu 的内部对象。通过菜单设计器可以设计菜单的 Items 属性，添加菜单项。MainMenu 组件在 Standard 组件面板中。

1. 菜单设计器

在窗体上添加 MainMenu 组件后，双击 MainMenu 组件，显示菜单设计器；或者用鼠标右键单击 MainMenu 组件，选择 Menu Designer... 也可以进入菜单设计器。如图 9-2 右边所示即为菜单设计器。

在菜单设计过程中，选择某个菜单项，则在 Object Inspector 对话框中显示的就是该菜单项所对应的属性。如图 9-2(b)所示，在菜单设计器中选择“打开”菜单项，则 Object



(a) Object Inspector 对话框　　(b) 菜单设计器

图 9-2　Object Inspector 对话框与菜单设计器

Inspector 对话框中显示的就是该菜单项的属性，如“打开”菜单项的 Name 属性是 N3，Caption 属性是“打开”。

菜单设计器有如下一些操作。

(1) 添加菜单项：一个菜单项设计完之后，自动在其后出现一个虚线框(如图 9-2(b)中“退出”菜单下面就是这种虚框)，表示新添加的菜单项。如果用户想添加新菜单项，用户可以直接进行设计。

(2) 移动菜单项：用鼠标将要移动的菜单项直接拖动到目的位置即可。

(3) 插入菜单项：用鼠标右键单击菜单项，选择 Insert，可以完成菜单项的插入操作。

(4) 删除当前菜单项：选中某菜单，直接按 Delete 键即可。

(5) 创建子菜单项：选中菜单，执行弹出式菜单中的 Create SubMenu 菜单。

菜单的属性如下。

(1) Caption 属性和 Name 属性：用于设置菜单的标题，如图 9-2(b)所示的“文件”、“新建”等。Name 属性默认值一般为 n1、n2、n3 等。

下面示例给出了 Caption 的一些用法，例如：

```
n2.Caption:='新建';          //菜单标题是新建,n2 是新建菜单的 Name 属性
n3.Caption:='打开[&O]';      //设置标题和加速键,按“Alt+O”等价于单击“打开”
n4.Caption:='-';             //设置菜单的分隔,用于分隔不同性质的菜单,以便更有条理
```

(2) Visible 属性：设置菜单是否可见，可以将菜单项设置为可见和不可见，还可以将菜单的子菜单设置为可见和不可见。

例如：假设 n1 是“文件”菜单的 Name 属性，n3 是“打开”菜单的 Name 属性。

```
n1.Visible:=False;           //“文件”菜单不显示,其下的“新建”、“打开”等菜单也不显示
n3.Visible:=False;           //“打开”菜单不显示
```

(3) Enabled 属性：菜单是否有效。

例如：假设 n1 是“文件”菜单的 Name 属性，n3 是“打开”菜单的 Name 属性。

```
n3.Enabled:=False;           //“打开”菜单无效
n1.Enabled:=False;           //整个“文件”菜单无效
```

说明：无效的菜单，无法用加速键或快捷键访问。

(4) ShortCut 属性：用于设置菜单的快捷键。

(5) Checked 属性和 AutoCheck 属性：Checked 属性用于设置菜单复选状态，值为

True 时，菜单旁边显示符号“√”。AutoCheck 属性用于设置菜单是否自动改变 Checked 属性值，当 AutoCheck 属性值为 True 时，单击菜单项，该菜单项的 Checked 属性值自动改变，否则需要用代码来改变。

例如：假设 n1、n2、n3、n4 分别表示“字体”、“粗体”、“斜体”和“下划线”菜单的 Name 属性，其中 n1 是顶层菜单，n2 的 AutoCheck 属性值为 False，n3 的 AutoCheck 属性值为 True。

```
n2.Checked:=not n2.Checked;
//该语句用于改变 n2 的 Checked,单击 n2 菜单项,它的 Checked 属性值不断改变。
//n3 菜单不需要编写代码,每次单击 n3,n3 的 Checked 属性值能自动改变。
```

(6) RadioItem 属性和 GroupIndex 属性：用于设置单选菜单项，值为 True 时表示该菜单为单选菜单项。如果多个菜单的 RadioItem 属性都是 True，且 GroupIndex 属性(整型，取值范围 0～255)相同，则这些菜单项是一组单选菜单，任何时刻仅可以选择其中一个菜单项。

2. 下拉菜单应用

【例 9-1】 设置一个编辑器，该编辑器可以打开、保存、新建文件，还可以编辑文件、设计字体。

分析：本章主要讲授菜单知识，因此对文件的新建、打开、保存等不做讲解。通过本题，主要讲授菜单的有效、单选菜单、复选菜单、加速键、菜单代码编写等。

步骤如下：

(1) 在窗体上添加下拉菜单组件 MainMenu1、添加打开对话框 OpenDialog1、添加保存对话框 SaveDialog1、添加多格式编辑框 RichEdit1。

(2) 设置菜单，菜单项如图 9-3 所示。其中“文件”、“新建”、“打开”、“保存”、分隔线和“退出”分别为 n1、n2、n3、n4、n5 和 n6。“编辑”、“复制”、“剪切”、“删除”和“粘贴”的 Name 属性分别为 n7、n8、n9、n10 和 n11。“字体”、“粗体”一直到“隶书”的 Name 属性分别为 n12～n20。设置各个菜单的加速键，此处省略。

(3) 设置 RichEdit 组件的 Align 属性为 alClient，调整这些组件的大小和位置，如图 9-3 所示。

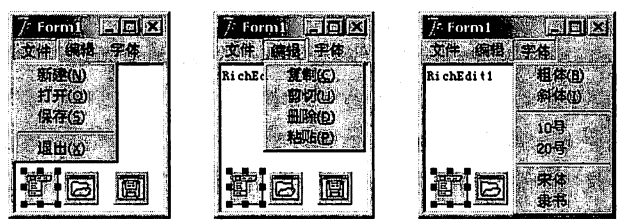


图 9-3 编辑器界面及其菜单项

(4) 设置“粗体”和“斜体”为复选菜单，方法是设置 n13 和 n14 的 AutoCheck 为 False。编写粗体和斜体的代码如下：

```
procedure TForm1.N13Click(Sender: TObject);
begin
 n13.Checked:=not n13.Checked;        //每次单击"粗体"均要改变粗体的 Checked 属性值
 if n13.Checked then
  richedit1.SelAttributes.Style:=richedit1.SelAttributes.Style+[fsbold]
 else
  richedit1.SelAttributes.Style:=richedit1.SelAttributes.Style-[fsbold];
end;
procedure TForm1.N14Click(Sender: TObject);
begin
 n14.Checked:=not n14.Checked;        //每次单击"斜体"均要改变斜体的 Checked 属性值
 if n14.Checked then
  richedit1.SelAttributes.Style:=richedit1.SelAttributes.Style+[fsitalic]
 else
  richedit1.SelAttributes.Style:=richedit1.SelAttributes.Style-[fsitalic]
end;
```

分析：由于 n13 和 n14 的 AutoCheck 属性为 False，因此单击"粗体"和"斜体"，菜单的 Checked 属性值不会自动改变，因此需要使用语句来改变 n13 和 n14 的 Checked 属性值。显然，如果将 n13 和 n14 的 AutoCheck 属性设置为 True，则上面的程序可以写成：

```
procedure TForm1.N13Click(Sender: TObject);
begin
 if n13.Checked then
  richedit1.SelAttributes.Style:=richedit1.SelAttributes.Style+[fsbold]
 else
  richedit1.SelAttributes.Style:=richedit1.SelAttributes.Style-[fsbold]
end;

procedure TForm1.N14Click(Sender: TObject);
begin
 if n14.Checked then
  richedit1.SelAttributes.Style:=richedit1.SelAttributes.Style+[fsitalic]
 else
  richedit1.SelAttributes.Style:=richedit1.SelAttributes.Style-[fsitalic]
end;
```

(5) 设置"10 号"、"20 号"、"宋体"和"隶书"为单选菜单。关于 AutoCheck 属性，这里不再赘述。为了使程序更加简洁，将这 4 个菜单项的 AutoCheck 属性都设置为 True。这 4 个菜单的 RadioItem 属性设置为 True，表示都是单选菜单。由于"10 号"和"20 号"是一组，故将"10 号"和"20 号"的 RadioIndex 设置为 0(RadioIndex 相同的为一组)，将另外一组菜单"宋体"和"隶书"的 RadioIndex 都设置为 1。编写这 4 个菜单的代码如下：

```
procedure TForm1.N16Click(Sender: TObject);
begin
 if n16.Checked then richedit1.SelAttributes.Size:=10
end;
```

```
procedure TForm1.N17Click(Sender: TObject);
begin
 if n17.Checked then richedit1.SelAttributes.Size:=20
end;

procedure TForm1.N19Click(Sender: TObject);
begin
 if n19.Checked then richedit1.SelAttributes.Name:='宋体'
end;

procedure TForm1.N20Click(Sender: TObject);
begin
 if n20.Checked then richedit1.SelAttributes.Name:='隶书'
end;
```

(6) 编写 RichEdit1 的 OnSelectionChange 事件过程如下：

```
procedure TForm1.RichEdit1SelectionChange(Sender: TObject);
begin
 if richedit1.SelLength>0 then
  begin
   n8.Enabled:=true;        //如果有选定文字,则设置"复制"、"剪切"和"删除"有效
   n9.Enabled:=true;
   n10.Enabled:=true
  end
 else
  begin
   n8.Enabled:=false;
   n9.Enabled:=false;
   n10.Enabled:=false
  end;
 if fsbold in richedit1.SelAttributes.Style then
  n13.Checked:=true         //选定文字是粗体,则设置 n13 的 Checked 为 True
 else n13.Checked:=false;
 if fsitalic in richedit1.SelAttributes.Style then
  n14.Checked:=true
 else n14.Checked:=false;
 if richedit1.SelAttributes.Size =10 then
  n16.Checked:=true;
 if richedit1.SelAttributes.Size =20 then
  n17.Checked:=true;        //选定"20 号",则"10 号"的 Checked 属性自动变为 False
 if richedit1.SelAttributes.Name='宋体' then
  n19.Checked:=true;
 if richedit1.SelAttributes.Name='隶书' then
  n20.Checked:=true;
end;
```

说明：该过程的作用是根据选定文字的状态设置菜单的状态。"文件"、"编辑"类菜单的代码请读者自己完成。

3. 菜单模板

我们可以像使用模具一样使用菜单模板来设计自己的菜单，也可以把自己设计的菜单保存为菜单模板，供下次使用。

(1) 添加系统菜单。

在自己的菜单中添加系统菜单模板，使得设计菜单更加方便、快捷。鼠标右键单击菜单项选择 Insert From Template，可以将系统中的菜单添加到自己的菜单中，如 File、Edit、Window、Help 等。

(2) 将自己的菜单保存为菜单模板。

鼠标右键单击菜单，选择 Save As Template，可以将自己设计的菜单保存为模板，以供下次使用。同样可以使用 Delete Template 删除不用的菜单模板。

9.1.2　弹出式菜单组件 PopupMenu

弹出式菜单与下拉式菜单类似。弹出式菜单是为某个组件设计的，只有在某个组件上单击鼠标右键才可以显示该菜单，因此必须将弹出式菜单与某个组件关联起来。方法是将弹出式菜单作为该组件的 PopupMenu 属性值。

9.1.3　在菜单中添加小图标

在菜单中添加图标，可以使菜单更加美观、形象，如图 9-4 所示。在菜单中添加图标需要使用到另外一个组件 ImageList。

1. ImageList 组件

ImageList 组件的作用是装载图像，图像可以供其他组件(如菜单、工具栏等)使用，ImageList 组件在 Standard 组件面板中。双击 ImageList 组件，激活 ImageList 编辑器，如图 9-5 所示。

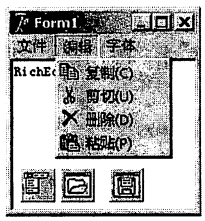


图 9-4　在菜单中添加图标

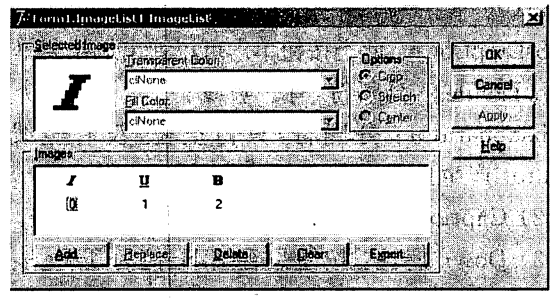


图 9-5　ImageList 编辑器

使用 Add、Delete、Clear 按钮，分别向 ImageList 中添加图像、删除图像或者清空全部图像。

2. ImageList 组件的使用

ImageList 的作用是为其他组件提供图像，我们以菜单为例讲述 ImageList 组件的使用方法。

(1) 整个菜单与 ImageList 组件关联。

设置菜单组件 MainMenu1 的 Images 属性为 ImageList1，使整个菜单与 ImageList 组件 ImageList1 关联。

(2) 设置菜单项与 ImageList 中的图像关联。

设置某个菜单项的 ImageIndex 值，则 ImageList 中的图像可以显示在该菜单项中。第 1 个图像的 ImageIndex 值为 0，而－1 表示在菜单项中不使用图标。

其他组件与 ImageList 关联的方法与此类似。

9.2 工具栏组件 ToolBar

在 Delphi 中工具栏是很重要的界面元素。一般把那些经常使用的菜单做成工具栏，使用户操作更加方便快捷，使用 Delphi 可以制作出水准很高的工具栏。工具栏如图 9-6 所示。在 Delphi 中既可以使用 SpeedButton 制作工具栏，也可以使用 ToolBar 组件制作工具栏。使用 SpeedButton 制作工具栏请读者自己完成。本节将介绍使用 ToolBar 制作工具栏。

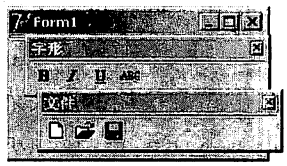


图 9-6 可以拖放和停靠的工具栏

ToolBar 组件位于 Win32 组件面板上，它带有自己的按钮 ToolButton，设置属性可以将按钮设计成各种不同的风格。

1. ToolBar 工具栏和 ToolButton 按钮

添加 ToolBar 到窗体上，鼠标右键单击 ToolBar，选择 New Button 或 New Separator 可以创建 ToolBar 按钮。按钮 ToolButton 的属性如下。

(1) Style 属性：用于设置按钮的样式。按钮的样式有如下取值。

tbsButton：一般的按钮。

tbsCheck：具有复选框功能的按钮；也可以与其他按钮一起形成一组单选按钮。

tbsDropDown：此类按钮单击可以弹出下拉菜单，下拉菜单由 DropdownMenu 属性指定。

tbsDivider：分隔符，有一竖线。

tbsSeparator：分隔符，只留空位，没有竖线。鼠标右键单击 ToolBar 选择 New Separator 产生此类分隔符。

(2) Caption 属性：按钮的标题文字。

(3) Down 属性：按钮是否按下，值为 True 表示按钮被按下，否则按钮为抬起状。

(4) ImageIndex 属性：按钮图标的序号，按钮图标来自 ImageList。在 ToolBar 中设置 Images 属性，可以设置 ImageList。

(5) Grouped 属性：值为 True 时按钮可与其他按钮形成一组单选按钮。在该组按钮中仅可以有一个按钮被按下。

(6) DropdownMenu 属性：决定按钮的下拉菜单。

按钮 ToolButton 的属性用于控制按钮的外观等属性，而工具栏 ToolBar 的属性则用于控制整个工具栏的外观样式，其属性如下。

(1) Align 属性：工具栏的位置。例如，值为 alTop 表示工具栏在窗体顶部。

(2) Flat 属性：值为 True 表示按钮的平面样式，否则按钮是立体样式。

(3) Images 属性：设置 ImageList，为 ToolButton 提供图标。

(4) ShowCaption 属性：值为 True 时在 ToolButton 上显示 Caption，否则不显示。

2. ToolBar 工具栏的应用

【例 9-2】 利用 ToolBar 制作一个工具栏，用于设置标签的字体。

步骤如下：

(1) 在窗体上添加 ToolBar 工具栏组件 ToolBar1、标签 Label1、弹出式菜单 PopupMenu1、图像列标组件 ImageList1 共 4 个。

(2) 在 PopupMenu1 中添加 3 个 Caption 分别为“10 号”、“20 号”和“30 号”的菜单项 n1、n2 和 n3，并把这 3 个菜单项设置为单选菜单，操作方法省略。在 ImageList1 中添加图标两个，如图 9-6(a)所示。

(3) 设置 ToolBar1 的属性，参见表 9-1。

表 9-1　ToolBar1 的属性

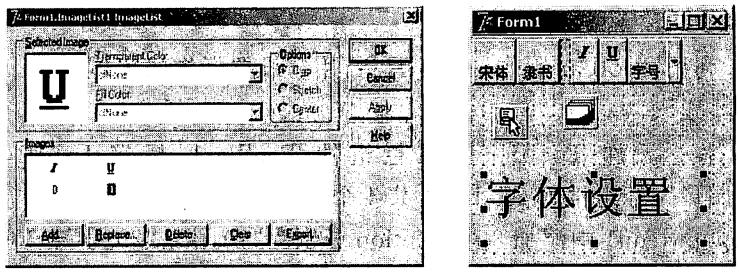
属　性	属 性 值	说　明
Align	alTop	工具栏在窗体顶部
Flat	False	立体按钮
ShowCaption	True	显示按钮的 Caption
Images	ImageList1	按钮的图标来自 ImageList1

(4) 为 ToolBar1 添加 6 个按钮，设置属性，参见表 9-2。

表 9-2　各个按钮的属性设置

对　象	属　性	属 性 值	说　明
ToolButton1、ToolButton2	Style	tbsCheck	ToolButton1 和 ToolButton2 是一组单选按钮
	Grouped	True	
	ImageIndex	−1	
	Caption	“宋体”和“隶书”	
ToolButton3	Style	tbsSeparator	分隔符
	ImageIndex	−1	无须图标
ToolButton4、ToolButton5	Style	tbsCheck	ToolButton4 和 ToolButton5 是复选按钮“斜体”和“下划线”
	Grouped	False	
	ImageIndex	0 和 1	
ToolButton6	Style	tbsDropDown	下拉菜单式按钮
	Grouped	False	无须分组
	DropdownMenu	PopupMenu1	指明下拉菜单
	ImageIndex	−1	无须图标
	Caption	“字体”	标题

至此，程序界面如图 9-7(b)所示。



(a) ImageList 组件对话框　　(b) 程序设计界面

图 9-7　ImageList 组件对话框和程序设计界面

(5) 编写代码如下：

```
procedure TForm1.ToolButton1Click(Sender: TObject);
begin
 label1.Font.Name:='宋体'
end;

procedure TForm1.ToolButton2Click(Sender: TObject);
begin
 label1.Font.name:='隶书'
end;

procedure TForm1.ToolButton4Click(Sender: TObject);
begin
 if toolbutton4.Down
 then label1.Font.Style:=label1.Font.Style+[fsitalic]
 else label1.Font.Style:=label1.Font.Style-[fsitalic]
end;

procedure TForm1.ToolButton5Click(Sender: TObject);
begin
 if toolbutton5.Down
 then label1.Font.Style:=label1.Font.Style+[fsunderline]
 else label1.Font.Style:=label1.Font.Style-[fsunderline]
end;

procedure TForm1.n1Click(Sender: TObject);
begin
 label1.Font.Size:=10
end;

procedure TForm1.n2Click(Sender: TObject);
begin
 label1.Font.Size:=20
end;
```

```
procedure TForm1.n3Click(Sender: TObject);
begin
 label1.Font.Size:=30
end;
```

(6) 按 F9 键运行程序，运行结果如图 9-8 所示。

3. 利用 CoolBar 组件和 ToolBar 组件制作工具栏

CoolBar 组件是一个容器组件，可以在 CoolBar 中添加多个单元格，每个单元格可以用来存放其他组件，如存放工具栏。CoolBar 组件在 Win32 组件面板中。

鼠标右键单击 CoolBar 组件，选择菜单 Bands Editor...，出现 Editing CoolBar1. Bands 对话框，可以添加单元格。鼠标右键单击 Editing CoolBar1. Bands 空白处选择 Add 可以为 CoolBar 添加单元格。如图 9-9 就是添加了两个单元格的 CoolBar。

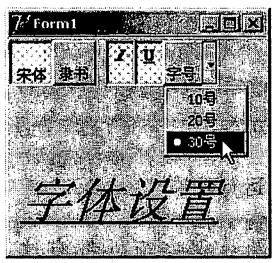


图 9-8　程序运行结果

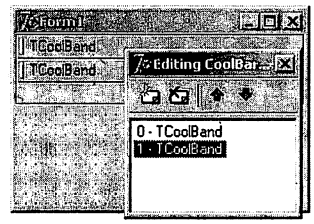


图 9-9　CoolBar 组件

CoolBar 组件的常用属性如下。

(1) AutoSize 属性：是否自动调节大小适应单元格中的组件。

(2) DockSite 属性：是否可以停靠其他组件，值为 True 时，可以接受停靠。

单元格的常用属性如下。

(1) Control 属性：单元格中存放的组件，如工具栏。

(2) Text 属性：单元格的文字。

【例 9-3】 利用 CoolBar 组件和 ToolBar 组件制作专业级的工具栏。

步骤如下：

(1) 添加 ToolBar 组件。在窗体上添加 ToolBar 组件 ToolBar1、ToolBar2 和图像列表组件 ImageList1。在 ToolBar1 中添加 3 个按钮，在 ToolBar2 中添加 4 个按钮。在 ImageList1 中添加 7 个图标。设置 ToolBar1、ToolBar2 和按钮的属性(此处省略)。设置属性后的界面如图 9-10 所示。

(2) 添加 CoolBar 组件。在窗体上添加 CoolBar 组件 CoolBar1。鼠标右键单击 CoolBar1 空白处，选择菜单 Bands Editor...，出现 Editing CoolBar1. Bands 对话框。鼠标右键单击 Editing CoolBar1. Bands 对话框空白处，选择 Add 为 CoolBar 添加两个单元格。界面如图 9-11 所示。

(3) 将 ToolBar 加入 CoolBar 单元格。设置 CoolBar1 第 0 单元格的 Control 属性为 ToolBar1，设置 CoolBar1 第 1 单元格的 Control 属性为 ToolBar2。将 CoolBar1 的 AutoSize

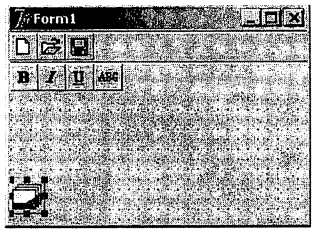


图 9-10 添加 ToolBar 组件的窗体

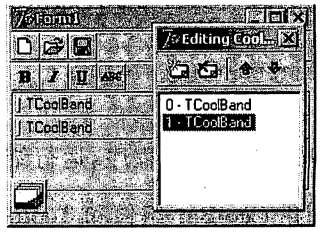
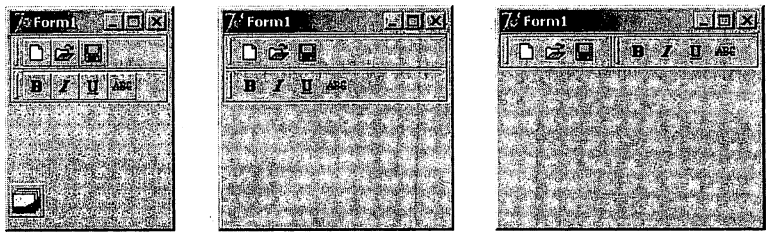


图 9-11 添加 CoolBar 组件的窗体

设置为 True。此时界面如图 9-12(a)所示。



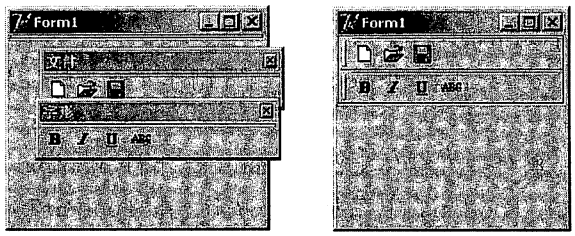
(a) (b) (c)

图 9-12 加入 CoolBar 中的工具栏以及运行效果

(4) 实现拖放和停靠的功能,必须设置 3 个重要的属性。很多组件都有这 3 个属性,读者应该会举一反三。

设置 ToolBar1 和 ToolBar2 的 DragKind 为 dkDock,表示 ToolBar1 和 ToolBar2 可以拖放;设置 ToolBar1 和 ToolBar2 的 DragMode 为 Automatic,表示使用鼠标自动拖放,无须使用代码。拖放后的工具栏如图 9-13(a)所示。

设置 CoolBar1 的 DockSite 属性为 True,表示 CoolBar 可以停靠。停靠后的界面如图 9-13(b)所示。



(a) (b)

图 9-13 可以拖放和停靠的工具栏

(5) 工具栏拖出来后,可以点击“关闭”按钮进行关闭,此时若执行语句“ToolBar1.Visible :=true”可以将工具栏再次显示出来。

说明:本程序主要讲述工具栏的制作,因此代码编写本程序没有涉及。另外,很多组件都有 DragMode、DockSite 和 DragKind 属性,请读者举一反三。

9.3　状态栏组件 StatusBar

和菜单、工具栏一样，状态栏也是 Delphi 中重要的界面元素。菜单和工具栏主要进行输入操作，而状态栏主要进行输出操作，用户根据状态栏的“状态”可以了解到程序执行情况。状态栏组件在 Win32 组件面板中。

StatusBar 有两种结构：单面板和多面板，由 SimplePanel 属性决定。

1. 单面板

当 StatusBar 的 SimplePanel 属性值为 True 时，状态栏是单面板，状态栏的文本内容由 SimpleText 属性决定。

2. 多面板

当 StatusBar 的 SimplePanel 属性值为 False 时，状态栏是多面板。多面板的每格都有自己的属性，可以使用 Editing StatusBar1. Panels 编辑器编辑状态栏的子面板(格)。双击 StatusBar 组件打开 Editing StatusBar1. Panels 编辑器。鼠标右键单击空白处选择 Add 添加状态栏的子面板(格)。如图 9-14 所示是添加了 3 个子面板的状态栏。

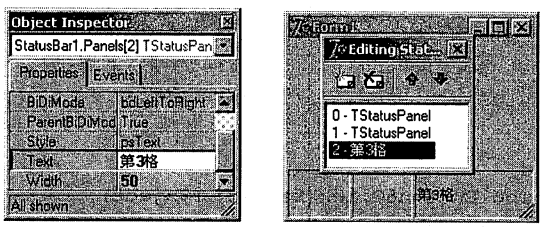


图 9-14　添加了 3 个子面板的状态栏及其属性设置

使用编辑器上提供的工具栏按钮，可以实现子面板的编辑功能。

Add New 按钮：用于添加子面板。

Delete Select 按钮：删除所选择的子面板。

Move Selected Up 按钮：将所选按钮向上移动。

Move Selected Down 按钮：将所选按钮向下移动。

3. 状态栏的应用

【例 9-4】 为例 9-2 制作状态栏。

步骤如下：

(1) 在窗体上添加状态栏。为状态栏添加 4 个子面板，添加一个计时器组件 Timer1。

(2) 为子面板设置 Text 属性值，分别为“字体：”、“字形：”、“大小：”和“时间：”，设置每个子面板的宽度为 80，如图 9-15 所示。

(3) 编写程序如下：

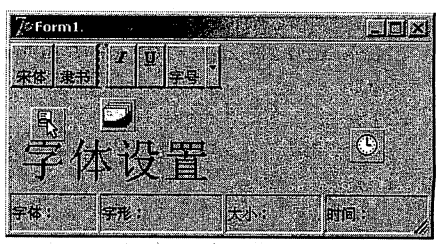


图 9-15　为例 9-2 制作状态栏

```
procedure TForm1.ToolButton1Click(Sender: TObject);
begin
 label1.Font.Name:='宋体';
 statusbar1.Panels[0].Text:='字体:宋体';                    //第 1 格
end;

procedure TForm1.ToolButton2Click(Sender: TObject);
begin
 label1.Font.name:='隶书';
 statusbar1.Panels[0].Text:='字体:隶书';
end;

procedure TForm1.ToolButton4Click(Sender: TObject);
begin
 if toolbutton4.Down
 then label1.Font.Style:=label1.Font.Style+[fsitalic]
 else label1.Font.Style:=label1.Font.Style-[fsitalic];
 if (fsitalic in label1.Font.Style) and (fsunderline in label1.Font.Style)
 then statusbar1.Panels[1].Text:='字形:斜体,下划线'       //第 2 格
 else if fsitalic in label1.Font.Style
      then statusbar1.Panels[1].Text:='字形:斜体'
      else if fsunderline in label1.Font.Style
           then statusbar1.Panels[1].Text:='字形:下划线'
           else statusbar1.Panels[1].Text:='字形:'
end;

procedure TForm1.ToolButton5Click(Sender: TObject);
begin
 if toolbutton5.Down
 then label1.Font.Style:=label1.Font.Style+[fsunderline]
 else label1.Font.Style:=label1.Font.Style-[fsunderline];
 if (fsitalic in label1.Font.Style) and (fsunderline in label1.Font.Style)
 then statusbar1.Panels[1].Text:='字形:斜体,下划线'
 else if fsitalic in label1.Font.Style
      then statusbar1.Panels[1].Text:='字形:斜体'
      else if fsunderline in label1.Font.Style
           then statusbar1.Panels[1].Text:='字形:下划线'
           else statusbar1.Panels[1].Text:='字形:'
end;

procedure TForm1.n1Click(Sender: TObject);
begin
 label1.Font.Size:=10;
 statusbar1.Panels[2].Text:='大小:10';                     //第 3 格
end;

procedure TForm1.n2Click(Sender: TObject);
begin
 label1.Font.Size:=20 ;
```

```
 statusbar1.Panels[2].Text:='大小:20';
end;

procedure TForm1.n3Click(Sender: TObject);
begin
 label1.Font.Size:=30;
 statusbar1.Panels[2].Text:='大小:30';
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
 statusbar1.Panels[3].Text:='时间:'+timetostr(now);     //第 4 格
end;
```

(4) 运行结果如图 9-16 所示。

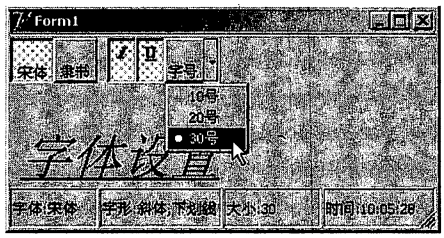


图 9-16　添加状态栏后的程序运行结果

9.4　小结

本章讲述了菜单、工具栏和状态栏，菜单、工具栏是本章的重点。读者应该学会制作多种风格和样式的菜单，例如弹出式菜单、下拉式菜单、含有小图标的菜单、复选菜单、单选菜单等。读者还应学会制作外观优美的工具栏，比如含有小图标的工具栏、可以拖曳和停靠的工具栏，以及不同样式按钮的工具栏。读者要学会使用单面板和多面板的状态栏。

习题

1. 怎样设置单选菜单？
2. 怎样设计复选菜单？
3. 怎样设计单选的工具栏？
4. 怎样设计复选的工具栏？
5. 为例 3-20 设置单面板状态栏，要求在状态栏上显示选中的唐诗。
6. 为例 3-4 添加状态栏，状态栏有 4 格，第 1 格提示对错，第 2 格给出答案，第 3 格给出做错的题数，第 4 格给出百分制的分数。
7. 利用 SpeedButton 设置工具栏。
8. 制作一个简单的编辑器，要求使用菜单、工具栏、状态栏。

第 10 章

图形图像与多媒体

Delphi 不仅提供了简洁有效的图形图像组件，还提供了用于绘制图形的方法。另外，Delphi 还提供了功能强大、使用灵活的多媒体组件。基于这些，使用 Delphi 开发图形图像程序和多媒体程序就变得轻松自如。

10.1 图形图像程序设计

10.1.1 图形组件 Shape

Shape 组件位于 additional 页上，用于在窗体上绘制一些简单的图形，如正方形、长方形、圆形等。Shape 组件具有如下属性。

(1) Shape 属性

Shape 属性用于设置图形的形状，其属性值以及含义如表 10-1 所示。

表 10-1 Shape 属性值及含义

属性值	图 形	属性值	图 形
stCircle	圆形	stRoundRect	圆角长方形
stEllipse	椭圆	stRoundSquare	圆角正方形
stRectangle	矩形	stSquare	正方形

图 10-1 更加形象直观地表示了 Shape 属性。

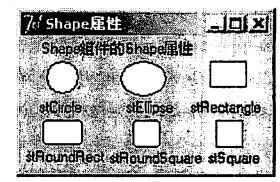


图 10-1 Shape 属性取值

(2) Brush 属性

Brush 属性用于设置图形的填充模式和颜色。它包含两个子属性：Color 和 Style，分别表示填充颜色和填充样式。

Color 子属性：决定图形内部的填充颜色。例如，clRed 表示红色，clBlue 表示蓝色，clSilve 表示银色等。

Style 子属性：表示图形的填充样式。其属性取值如图 10-2 所示。

(3) Pen 属性

Pen 属性用于设置图形的边框。它具有几个子属性，如 Color、Mode、Style 和 Width。

Color 子属性：用于设置图形边框的颜色。例如，clRed 表示红色，clBlue 表示蓝色，

clSilve 表示银色等。

Mode 子属性：决定画笔是如何画成的。

Style 子属性：决定边框线条。其属性取值如图 10-3 所示。

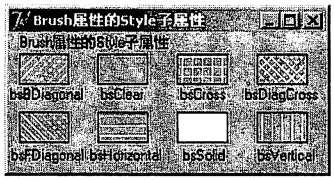


图 10-2　Brush 属性的 Style 子属性

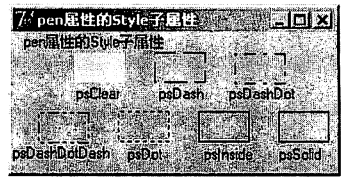


图 10-3　Pen 属性的 Style 子属性

(4) Width 属性

Width 属性表示边框的线宽，默认值是 1。

10.1.2　图像组件 Image

Image 组件用于显示图像文件，要显示的图像文件在其 Picture 属性中指定。Image 组件在 Additional 组件面板中。

Image 组件的属性如下。

(1) Picture 属性：通过设置该属性，可以在程序运行初始阶段在 Image 组件中装入一个图像文件，文件类型可以是 bmp、jpg、jpeg、wmf、emf 和 ico 等。也可以在程序运行期间通过语句载入一幅 bmp 格式的图片，例如：image1. Picture. LoadFromFile('d:\garden . bmp')。

(2) AutoSize 属性：Image 是否自动改变大小以适应图像文件的尺寸。值为 True 时，图像框自动改变以适应图像文件；值为 False 时，图像框不随图像文件的改变而改变。

(3) Stretch 属性：该属性设置为 True 时，图像自动缩放以填满整个图像框。

【例 10-1】 在窗体上添加图像组件 Image1，添加命令按钮 Button1。单击 Button1 显示一幅图片。

设计步骤：

(1) 添加组件，设置属性，界面如图 10-4 所示。

(2) 编写程序如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 image1.Picture.LoadFromFile('view.bmp');
end;
```

(3) 运行程序，发现只能够显示图像左上角一部分，如图 10-5 所示。

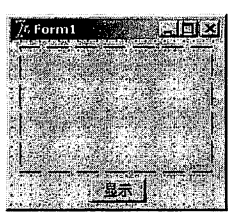


图 10-4　程序界面

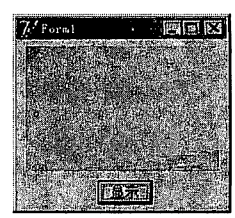


图 10-5　autosize＝False stretch＝False

要想显示图像的全部，有两种解决方法。第一种是设置属性 stretch 值为 True，这样图像可以拉伸或者压缩，此时可以看见全图，如图 10-6 所示。第二种解决方法是将 autosize 设置为 True，这时图像框自动变大或者缩小，此时可以看到全图，如图 10-7 所示。

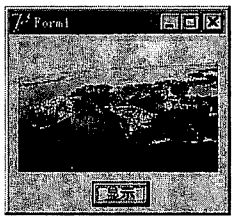


图 10-6 autosize=False stretch=True

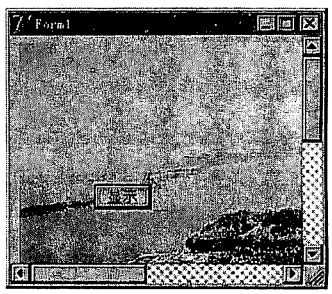


图 10-7 autosize=True 时的图像

10.1.3 画布 Canvas

画布 Canvas 对象用于应用程序的图形输入，在组件面板上看不到画布 Canvas。它有如下属性。

(1) Brush 属性：用于设置图形的填充部分。它包含 Color、Style 子属性，用于设置填充颜色和填充图案。用法与 Shape 组件类似。

(2) Font 属性：用于设置 Canvas 对象中字的大小(Size)、颜色(Color)和样式(Style)。

(3) Pen 属性：用于设置图形边框或线条。包括线条颜色、粗细、线条种类。它与 Shape 组件的 Pen 属性及其子属性类似。

(4) Pixels 属性：点，图形的最小单位。每个点都有 x 和 y 坐标。

1. 绘制点

绘制点的格式为：

```
Canvas. Pixels[x 坐标,y 坐标]:=点的颜色;
```

【例 10-2】 绘制多个点可以构成一些规则的图形，如直线、椭圆、圆等。

设计步骤如下：

(1) 添加按钮组件 Button1～Button4，设置属性。

(2) 绘制直线。编写 Button1 的 OnClick 事件过程如下：

```
procedure TForm1.Button1Click(Sender: TObject);

 var y:integer;

begin
 for y:=20 to 70 do
  //这是一条由点构成的垂直的直线段,x 坐标为 40,y 坐标从 20 到 70
  canvas.Pixels[40,y]:=clgreen;
end;
```

(3) 绘制圆。编写 Button2 的 OnClick 事件过程如下：

```
procedure TForm1.Button2Click(Sender: TObject);

 var i,x,y:integer;

begin
 for i:=1 to 360 do                          //利用参数方程的知识
  begin
   x:=round(100+30 * sin(i * pi/180));       //圆心为(100,50),半径为 30 的圆
   y:=round(50+30 * cos(i * pi/180));
   canvas.Pixels[x,y]:=clred;
  end;
end;
```

(4) 绘制椭圆。编写 Button3 的 OnClick 事件过程如下：

```
procedure TForm1.Button3Click(Sender: TObject);

 var i,x,y:integer;
begin
 for i:=1 to 360 do
  begin
   x:=round(150+50 * sin(i * pi/180));
   y:=round(50+30 * cos(i * pi/180));
   canvas.Pixels[x,y]:=clblue;
  //绘制椭圆:((x-150)/50)²+((y-50)/30)²=1
  end;
end;
```

(5) 绘制圆柱。编写 Button4 的 OnClick 事件过程如下：

```
procedure TForm1.Button4Click(Sender: TObject);
 var i,x,y,delta:integer;
begin
 for delta:=1 to 60 do                       //多个形状相同 y 坐标不一样的椭圆形成圆柱
  for i:=1 to 360 do                         //绘制椭圆
   begin
    x:=round(250+30 * sin(i * pi/180));
    y:=round(80+10 * cos(i * pi/180));
    canvas.Pixels[x,y-delta]:=clyellow;
   end;
 for i:=1 to 360 do                          //绘制圆柱最上面的椭圆
  begin
   x:=round(250+30 * sin(i * pi/180));
   y:=round(20+10 * cos(i * pi/180));
   canvas.Pixels[x,y]:=clgreen;
  end;
end;
```

(6) 运行程序,结果如图 10-8 所示。

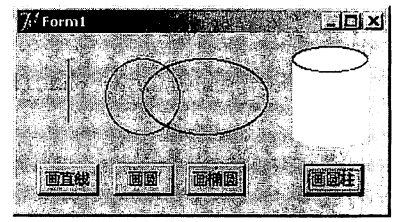


图 10-8 利用点绘制图形

2. 绘制直线

绘制直线的格式为:

```
Canvas.Moveto(起始点 x 坐标,起始点 y 坐标);
Canvas.Lineto(终止点 x 坐标,终止点 y 坐标);
```

例如:

```
canvas.moveto(10,20);
canvas.lineto(40,50);
```

从(10,20)到(40,50)画线段。

3. 绘制折线

绘制折线的格式为:

```
Canvs.PolyLine(point array of Tpoint);
```

例如:

```
var
 a:array [1..5]of Tpoint;

procedure TForm1.Button2Click(Sender: TObject);
begin
 a[1].x:=10; a[1].y:=10;
 a[2].x:=20; a[2].y:=30;
 a[3].x:=40; a[3].y:=10;
 a[4].x:=50; a[4].y:=20;
 a[5].x:=60; a[5].y:=10;
 canvas.Polyline(a);
end;
```

也可以写成如下格式:

```
canvas.Polyline([a[1],a[2],a[3]]);
```

4. 绘制多边形

绘制多边形的格式为:

```
Canvs. Polygon (point array of Tpoint);
```

方法与绘制折线类似,只是首尾点连接起来形成封闭的多边形。

5. 绘制圆和椭圆

绘制椭圆的格式为:

```
Canvs.Ellipse(x1,y1,x2,y2);
```

Canvs. Ellipse(x1,y1,x2,y2)绘制的椭圆与矩形(x1,y1,x2,y2)相切,其中(x1,y1)

和(x2,y2)表示矩形的左上角和右下角的点。当矩形呈正方形时绘制的就是圆。

画布还提供了绘制弧形曲线和矩形、圆角矩形的方法，这里不再赘述，感兴趣的读者可以参考相关书籍。

【例 10-3】 绘制各种图形。

程序如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 with canvas do
 begin
  MoveTo(10,20);
  LineTo(40,40);
  pen.Color:=clred;
  Polyline([point(50,10),point(60,40),point(70,5),point(80,20)]);
  Pen.Color:=clgreen;
  Rectangle(90,10,120,40);
  Polygon([point(150,25),point(160,40),point(130,20),point(170,5)]);
  Pen.Width:=1;
  Pen.Style:=psdash;
  Ellipse(160,20,250,100);
 end;
end;
```

程序运行结果如图 10-9 所示。

实际上，上述绘制的图形只要最小化窗体或在窗体被覆盖后，图形就会立即消失，如果想重新显示图形，需要重新绘制。程序如下：

```
procedure TForm1.FormPaint(Sender: TObject);
begin
 Button1Click(Sender)
end;
```

说明：最小化、最大化、窗体改变大小、窗体被遮盖都会触发 OnPaint 事件，所以此程序是在 Form1 的 OnPaint 事件过程中重新调用 Button1 的 OnClick 事件过程。重新绘制的图形如图 10-10 所示。

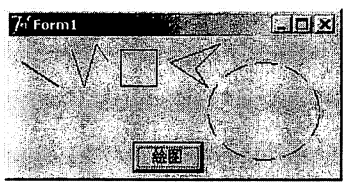


图 10-9　绘制各种图形

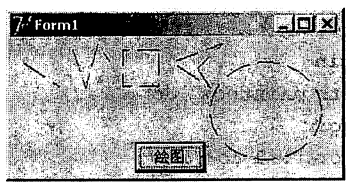


图 10-10　重绘后的图形

细心的读者一定会发现两次绘制的图形不完全一样，这是因为第一次执行了语句 Pen. Style :=psdash，该语句在第二次绘制图形的最开始就发生了作用。

10.1.4 画板组件 PaintBox

PaintBox 是一个用于绘制图形的矩形区域，该组件在 System 面板上。用户可以使用绘图语句在这个区域绘制各种图形。

【例 10-4】 将例 10-2 中的图形绘制在 PaintBox 中。

设计步骤如下：

(1) 在窗体上添加 PainBox 组件 PainBox1，添加 4 个按钮。调整组件的位置和大小，其中 PaintBox1 的 top=1，left=0。

(2) 编写代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);

 var y:integer;

begin
 with paintbox1 do
 for y:=20 to 70 do
  canvas.Pixels[40,y]:=clgreen;
end;
procedure TForm1.Button2Click(Sender: TObject);

 var i,x,y:integer;

begin
 with paintbox1 do
 for i:=1 to 360 do
  begin
   x:=round(100+30 * sin(i * pi/180));
   y:=round(50+30 * cos(i * pi/180));
   canvas.Pixels[x,y]:=clred;
  end;
end;

procedure TForm1.Button3Click(Sender: TObject);

 var i,x,y:integer;

begin
 with paintbox1 do
 begin
  for i:=1 to 360 do
   begin
    x:=round(150+50 * sin(i * pi/180));
    y:=round(50+30 * cos(i * pi/180));
    canvas.Pixels[x,y]:=clblue;
   end;
  end;
end;
```

```
procedure TForm1.Button4Click(Sender: TObject);

 var i,x,y,delta,r:integer;

begin
 with paintbox1 do
 begin
  for delta:=1 to 60 do
   for i:=1 to 360 do
    begin
     x:=round(250+30 * sin(i * pi/180));
     y:=round(80+10 * cos(i * pi/180));
     canvas.Pixels[x,y-delta]:=clyellow;
    end;
  for i:=1 to 360 do
   begin
    x:=round(250+30 * sin(i * pi/180));
    y:=round(20+10 * cos(i * pi/180));
    canvas.Pixels[x,y]:=clgreen;
   end;
 end;
end;

procedure TForm1.Button5Click(Sender: TObject);
begin
 paintbox1.Refresh        //刷新后,图形消失
end;
```

(3) 运行程序,结果如图 10-11 所示。

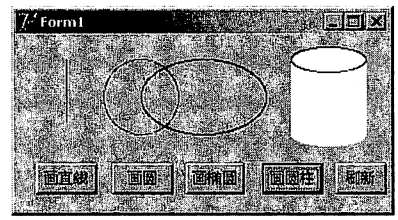


图 10-11　在 PainBox 上绘制图形

10.2　多媒体程序设计

Delphi 提供 Animate 和 MediaPlayer 两个组件,支持 AVI 影片、CD 唱片、MIDI 音乐、DAT 文件、WAVE 文件等格式。

10.2.1　Animate 组件

Animate 组件位于 Win32 组件面板中,用于播放 AVI 文件。不管 AVI 是否有声音,播放的时候都没有声音。

1. Animate 组件的常用属性

(1) Active 属性:值为 True 时可以播放,值为 False 时不可以播放。

(2) CommandAVI 属性:控制 Animate 组件播放保存于 Windows 的 Shell32.dll 中的内部图像序列,共有 9 种取值,如表 10-2 所示。相应的图像如图 10-12 所示。

表 10-2 CommandAVI 属性取值

取 值	说 明	取 值	说 明
aviNone	不显示系统图像	aviCopyFiles	复制多个文件图像
aviFindFolder	查找文件夹图像	aviRecycleFile	删除到回收站的图像
aviFindFile	查找文件图像	aviEmptyRecycle	清空回收站图像
aviFindComputer	查找计算机图像	aviDeleteFile	删除文件图像
aviCopyFile	复制文件图像		

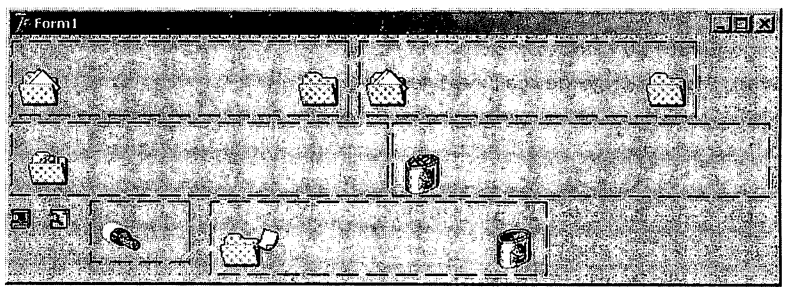


图 10-12 CommandAVI 属性取值以及所对应的图像

(3) FileName 属性：指定播放的 AVI 文件名。如果指定了 FileName 属性，则 CommandAVI 属性自动为 aviNone。

(4) StartFrame 属性和 StopFrame 属性：指重复播放时图像的起始和终止帧的序号。StartFrame 属性默认值为 1，StopFrame 属性的默认值为 FrameCount。

(5) FrameCount 属性：图像序列的总帧数。

(6) Repetitions 属性：重复播放的次数，值为 0 时表示无限播放。

(7) FrameWidth 属性和 FrameHeight 属性：图像序列的宽度和高度。

2. Animate 组件的常用方法

(1) Play 方法：格式为 procedure Play(FromFrame, ToFrame: Word; Count: Integer)，3 个参数分别为播放的起始帧数和终止帧数。

(2) Rest 方法：恢复 Animate 组件的默认值，显示第 1 帧，并置 Activate 属性值为 False。

(3) Seek 方法：格式为 procedure Seek(Frame: SmallInt)，参数为 Animate 组件要显示的帧。

(4) Stop 方法：停止播放，等价于设置 Activate 属性值为 False。

【例 10-5】 播放动画图片。

设计步骤如下：

(1) 添加组件，调整组件的大小和位置。界面如图 10-13 所示。

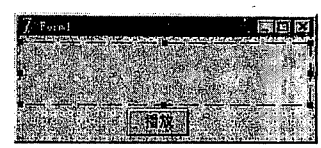


图 10-13 例 10-5 程序界面

(2) 设置 Animate1 的 CommandAVI 属性为

aviCopyFiles。

(3) 编写代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 if button1.Caption='播放'then
  begin
   animate1.Active:=true;
   button1.Caption:='停止'
  end
 else
  begin
   animate1.Seek(10);
   animate1.Active:=false;
   button1.Caption:='播放'
  end;

end;
```

(4) 程序运行结果如图 10-14 所示。

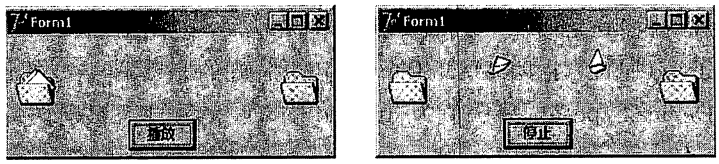


图 10-14　播放动画图片

10.2.2　媒体播放器组件 MediaPlayer

MediaPlayer 组件位于 system 组件面板中，其功能比 Animate 组件强大，不仅可以播放 AVI 文件，还可以播放其他多种格式的文件。

1. MediaPlayer 组件的按钮

MediaPlayer 组件具有一组按钮，外观与现实的播放器类似，如图 10-15 所示。

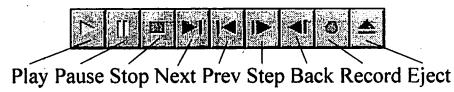


图 10-15　MediaPlayer 组件按钮

MediaPlayer 组件按钮功能参见表 10-3。

2. MediaPlayer 组件的常用属性

(1) AutoEnabled 属性：允许/禁止 MediaPlayer 组件的某些按钮。其值为 True 时，程序根据当前实际情况而允许/禁止某些按钮。其值为 False 时，按钮的允许/禁止通过设置 EnabledButtons 属性来控制。

(2) AutoOpen 属性：是否自动打开 FileOpen 属性所指定的文件。其值为 True 时，可以自动打开 FileName 属性所指的文件；否则需要使用 Open 语句来打开文件，默认值为 False。

表 10-3 MediaPlayer 按钮功能一览表

按钮名称	功　能
Play	播放媒体
Pause	暂停播放
Stop	停止播放或录制
Next	跳到下一个轨道，若不使用轨道则直接跳到媒体最后
Prev	跳到前一个轨道，若不使用轨道则直接跳到媒体最前面
Step	前进若干帧
Back	后退若干帧
Record	开始录制
Eject	退出媒体文件

(3) AutoRewind 属性：设置是否自动返回。其值为 True 时，当媒体播放完毕，将自动重置文件指针，此时按 Play 按钮可以重新播放；否则，必须按 Prev 按钮或调用 Previous 方法将指针移到起始点才可以重新播放。

(4) DeviceType 属性：指明 MediaPlayer 组件播放的是哪种类型的设备或文件，该属性属于 TMPDeviceType 类，默认值为 dtAutoSelect，表示自动识别设备类型。其属性值及含义参见表 10-4。

表 10-4 DeviceType 属性取值及含义

取　值	含　义	取　值	含　义
dtAutoSelect	设备类型由 FileName 的文件扩展名决定	dtOverlay	视频覆盖设备
dtAVIVideo	AVI 图像序列设备	dtScanner	扫描仪设备
dtCDAudio	CD 音频设备	dtSequencer	MIDI 音序器设备
dtDAT	数字音频磁带	dtVCR	录像机设备
dtDigitalVideo	数字视频设备	dtVideoDisc	视频设备
dtOthers	未定义设备	dtWaveAudio	Wave 音频设备

(5) Display 属性：为多媒体指定一个用于显示的组件，默认值为 nil 表示设备自动创建新的窗口来显示输出。

(6) FileName 属性：指定要播放的多媒体文件，是 String 类型。

(7) Mode 属性：只读属性，返回媒体设备的当前状态，TMPModes 类，取值及含义参见表 10-5。一般情况下在组件的 OnNotify 事件过程中检查该属性。

表 10-5　Mode 属性取值及含义

取　值	含　义	取　值	含　义
mpNotReady	设备未准备好状态	mpSeeking	定位状态
mpStopped	停止状态	mpPaused	暂停状态
mpPlaying	播放状态	mpOpen	设备打开状态
mpRecording	录制状态		

(8) Frames 属性：设置 step 操作时前进或者后退的画面帧数，Longint 类型。默认情况下当打开一个媒体文件时，Frames 就被设置为媒体文件帧数的 10%。

(9) EnabledButtons 属性：决定组件上的按钮可用或不可用。

(10) VisibleButtons 属性：用于设置按钮显示或不显示。

3. MediaPlayer 组件的常用方法

(1) Open 方法：打开媒体播放设备。每次播放前，必须打开媒体设备。

(2) Save 方法：将当前的数据保存到 FileName 指定的文件中。

(3) Play、Pause、Next、Previous、Step、Back、StartRecording、Eject 方法：功能参见表 10-3。

(4) Resume 方法：让媒体继续播放或继续录制。一般在暂停之后可以调用该方法。

(5) PauseOnly 方法：让媒体从播放或录制状态转为暂停状态。如果设备本来是暂停状态，则继续保持暂停状态，这一点与按钮 Pause 不一样。

【例 10-6】 媒体播放器程序。

设计步骤如下：

(1) 添加组件，设置属性(略)，界面如图 10-16 所示。

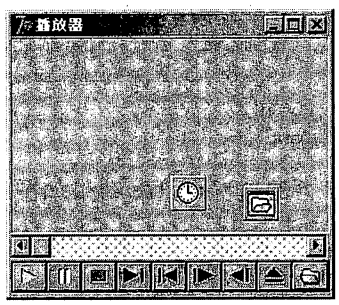


图 10-16　程序界面

(2) 编写代码如下：

编写 BitBtn1 的 OnClick 事件过程，完成打开文件操作。

```
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
 opendialog1.Filter:='*.avi|*.avi|*.*|*.*';
 if opendialog1.Execute then
  begin
   mediaplayer1.FileName:=opendialog1.FileName;
   mediaplayer1.Open;
   scrollbar1.Max:=mediaplayer1.Length;      //滚动条的最大值与媒体的长度相等
   timer1.Enabled:=true;                     //计时器开始计时
  end;
end;
```

程序运行开始执行的代码：

```
procedure TForm1.FormCreate(Sender: TObject);
```

```
begin
 mediaplayer1.Display:=panel1;                    //播放显示在 Panel1
 scrollbar1.Position:=0;                          //滚动条位置
end;
```

计时器用于控制滚动条的位置，使之与播放媒体的位置保持一致，代码如下：

```
procedure TForm1.Timer1Timer(Sender: TObject);
begin
 scrollbar1.Position:=mediaplayer1.Position;
end;
```

拖动滚动条可以改变当前播放媒体的位置，代码为：

```
procedure TForm1.ScrollBar1Scroll(Sender: TObject; ScrollCode: TScrollCode;
  var ScrollPos: Integer);
begin
 mediaplayer1.Pause;
 mediaplayer1.Position:=scrollbar1.Position;
    mediaplayer1.Play;
end;
```

播放完文件，媒体停止，代码如下：

```
procedure TForm1.MediaPlayer1Notify(Sender: TObject);
begin
 if mediaplayer1.Position=mediaplayer1.Length then
 begin
  mediaplayer1.Stop;
  timer1.Enabled:=false;
  scrollbar1.Position:=0;
 end;
end;
```

(3) 运行程序，结果如图 10-17 所示。

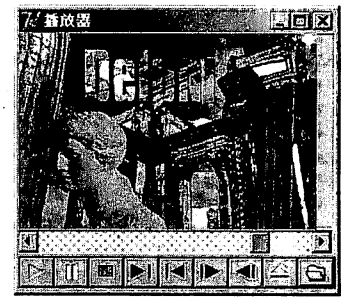


图 10-17 运行结果

10.3 小结

本章讲述了图形图像的基本组件，其中 Image 组件和 Canvas 组件非常重要。本章还讲述了多媒体程序设计，介绍了 Animate 组件和 MediaPlayer 组件。通过本章的学习，读者应该学会使用图形图像和多媒体组件编写应用程序。

教师和读者可以根据教学进程和课时实际情况，对本章进行适当的删减。

习题

1. 使用 Animate 组件编写动画播放器。
2. 使用 MediaPlayer 编写一个多媒体播放器。

3. 分别使用 Pixels 属性和 Ellipse 方法绘制相同形状的椭圆。
4. 在矩形区域(0,0,150,300)内绘制 10 个随机的 n 边形(n 不大于 6)。
5. 编写程序,产生如图 10-18 所示的圆柱体图形。
6. 请输入圆心和半径,在 PaintBox 中绘制圆形,如图 10-19 所示。

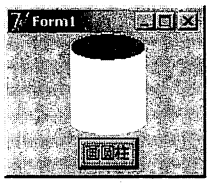


图 10-18　圆柱体图形

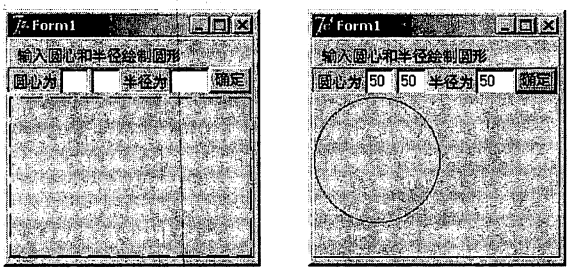


图 10-19　绘制圆形

7. 绘制若干个同心圆,使得整个圆变成实心圆,如图 10-20 所示。
8. 在窗体上绘制起始点、终止点随机的若干条直线段,颜色随机。如图 10-21 所示。

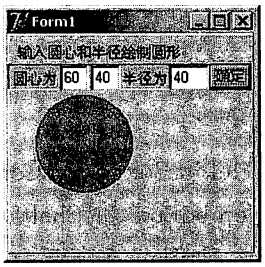


图 10-20　绘制实心的圆形

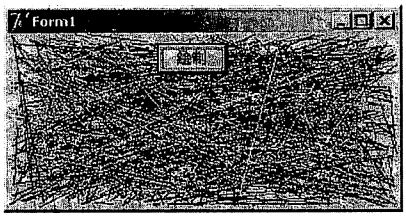


图 10-21　绘制随机直线段

第 11 章

文件编程

文件是同一类型元素的有序集合，是内存与外设之间传输数据的渠道。一些外设（如显示器、键盘和打印机）也可以看成是文件，但是最常见的文件是磁盘文件。本章讨论的文件专指磁盘文件。

Delphi 除继承了 Object Pascal 强大的文件管理功能之外，还提供了一些有用的组件用于管理文件。

11.1 文件管理组件

在 Delphi 7 中，用于文件和文件夹管理的组件有 FileListBox、DriveComboBox、DirectoryListBox、FilterComboBox、ShellTreeView、ShellComboBox 和 ShellListView。上述组件分别位于 Win 3.1 组件面板和 Sample 组件面板上。如图 11-1 所示。

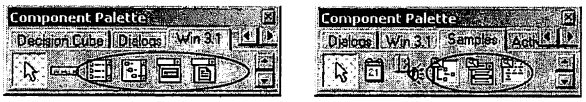


图 11-1 文件及文件夹管理组件

1. FileListBox 组件

该组件在 Win 3.1 组件面板中。该组件用于显示文件，其主要属性如下。

（1）Directory 属性：返回当前的文件目录，字符串类型。

（2）Drive 属性：用于设置或者返回当前的驱动器，字符型。例如：filelistbox1.Drive :='c'不能写成 filelistbox1.Drive :='c:'。

（3）FileEdit 属性：用于将当前选中的文件显示在文本框中。

（4）FileName 属性：显示文件列表框中当前被选中的文件，字符串型，含文件路径和文件名。

（5）FileType 属性：文件的类型，如只读文件、隐藏文件、系统文件以及是否显示子目录等。

（6）Mask 属性：用于设置文件列表框中显示文件的类型。例如：FileListBox1.

Mask := '*.exe'。

(7) ShowGlyphs 属性：用于设置文件列表中是否显示文件图标。值为 True 时，则显示文件图标；否则不显示文件图标，默认值为 False。

2. DirectoryListBox 组件

该组件用于显示指定驱动器下的文件夹列表，该组件在 Win 3.1 组件面板中。主要属性如下。

(1) Directory 属性：用于设置或者返回当前的文件目录。

(2) DirLabel 属性：Tlabel 型，用于将当前的文件目录显示在标签中。

(3) Drive 属性：用于设置或者返回当前的驱动器盘符，字符型。

(4) FileList 属性：TFileListBox 型，用于将文件夹列表框中选定文件夹中的文件显示在文件列表框中。

3. DriveComboBox 组件

该组件用于显示一可选驱动器下拉列表，供用户选择。该组件在 Win 3.1 组件面板中，主要属性如下。

(1) DirList 属性：用于将驱动器列表框连接到目录列表框。如果驱动器列表框选择发生改变，则目录列表框自动更新。

(2) Drive 属性：用于设置或者返回驱动器列表框的驱动器盘符。

(3) TextCase 属性：驱动器盘号是大写还是小写。tcLowerCase 表示小写，tcUpperCase 表示大写。

4. FilterComboBox 组件

FilterComboBox 组件用于显示一可选过滤器下拉列表，供用户选择。该组件在 Win 3.1 组件面板中，主要属性如下。

(1) FileList 属性：用于将本组件与文件列表框组件关联。如果当前文件类型改变，则文件列表框自动更新。

(2) Filter 属性：用于设置各种过滤文件类型。

(3) Mask 属性：用于存放所选过滤类型对应的字符串。

【例 11-1】 在窗体上添加 DriveComboBox 组件 DriveComboBox1、DirectoryListBox 组件 DirectoryListBox1、FileListBox 组件 FileListBox1、FilterComboBox 组件 FilterComboBox1、TLabel 组件 Label1 和 TEdit 组件 Edit1。

设计步骤如下：

(1) 添加组件，调整组件大小和位置，如图 11-2 所示。

(2) 设置各个组件的属性，见表 11-1。

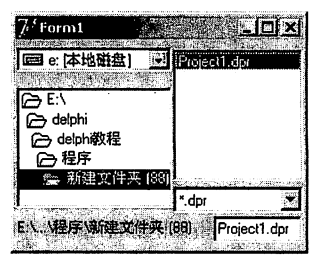


图 11-2　程序界面

表 11-1 各组件的属性设置

对 象	属 性	属 性 值	说 明
DriveComboBox1	DirList	DirectoryListBox1	DriveComboBox1 盘符改变，DirectoryListBox1 自动更新
DirectoryListBox1	Filelist	FileListBox1	FileListBox1 内容随 DirectoryListBox1 所选目录的改变而改变
	DirLabel	Label1	在 Label1 中显示所选择的目录
FileListBox1	FileEdit	Edit1	在 Edit1 中显示所选文件
FilterComboBox1	FileList	FileListbox1	FileListBox1 的内容随 FilterComboBox 所选文件类型的改变而改变
	Filter	如图 11-3 所示	设置在 FileListBox 中显示的文件类型

5. ShellTreeView 组件

ShellTreeView 组件可以通过树状结构实现对路径的浏览显示功能。该组件位于 Samples 组件面板中，常用属性如下。

(1) Root 属性：用于设置根目录。单击 Root 属性之后的 ，出现如图 11-4 所示的对话框，在对话框中用户可以选择不同的 Root 值。如果选择 Use Standard Folder 单选项，则 Root 属性值参见表 11-2。如果选择 Use Path 单选项，则用户可以在编辑框中直接输入路径，或者单击后面的按钮选择文件夹。

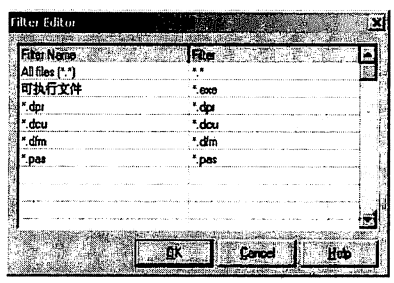


图 11-3 FilterComboBox 组件的 Filter 属性

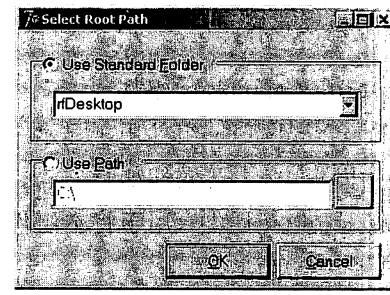


图 11-4 Root 属性对话框

表 11-2 Root 属性取值及含义

属 性 值	含 义
rfDesktop	Windows 桌面
rfMyComputer	Windows 系统中的“我的电脑”
rfNetWork	Windows 系统中的“网上邻居”
rfRecycleBin	Windows 系统中的“回收站”
rfControlPanel	Windows 系统中的“控制面板”
rfPrinter	Windows 系统中的“打印机”
rfProgram	Windows 系统中的“程序文件夹”

(2) PopupMenu 属性：当在树状文件或文件夹中单击鼠标右键时，将显示出弹出式菜单。

(3) ShellComboBox 属性：用于设置与当前 ShellTreeView 组件相关联的 ShellComboBox 组件。当 ShellComboBox 组件所指定的位置发生变化时，当前 ShellTreeView 组件的内容也发生相应变化。

(4) ShellListView 属性：用于设置与 ShellTreeView 组件相关联的 ShellListView 组件。当 ShellTreeView 组件选择不同的内容时，相应的 ShellListView 组件显示内容也随之改变。

6. ShellComboBox 组件

ShellComboBox 组件用于显示一个可选组合框，供用户选择路径。该组件位于 Sample 组件面板中，常用属性如下。

(1) Root 属性：用于设置根目录，它与 ShellTreeView 组件的 Root 属性基本相同。当 ShellComboBox 组件与 ShellTreeView 组件发生关联后，任何一个组件的 Root 属性改变，则另外一个组件的 Root 属性值也随之改变。

(2) ShellTreeView 属性：指定与 ShellComboBox 组件相关联的 ShellTreeView 组件。

(3) ShellListView 属性：指定与 ShellComboBox 组件相关联的 ShellListView 组件。

7. ShellListView 组件

ShellListView 组件用于实现在窗口中浏览文件或文件夹。该组件在 Samples 组件面板中，主要属性如下。

(1) Root 属性：设置根目录。用法与前面类似。

(2) ShellTreeView 属性：设置与 ShellListView 组件相关联的 ShellTreeView 组件。

(3) ShellComboBox 属性：设置与 ShellListView 组件相关联的 ShellComboBox 组件。

【例 11-2】 设计一个简单的资源管理器。

设计步骤如下：

(1) 在窗体上添加组件。

① 添加面板 Panel 组件 Panel1，设置 Panel1 的 Align 属性为 alTop，添加分隔条 Splitter 组件 Splitter1，设置 Splitter1 的 Align 属性为 alTop，Height 为 2。

② 在空白处添加面板 Panel 组件 Panel2，设置 Panel2 的 Align 属性为 alLeft，在空白处添加分隔条 Splitter 组件 Splitter2，设置 Splitter2 的 Align 属性为 alLeft，Width 为 2。

③ 在空白处添加面板 Panel 组件 Panel3，设置 Panel3 的 Align 属性为 alClient。

④ 在 Panel1 中添加 ShellComboBox 组件 ShellComboBox1，调整大小和位置到合适点。

⑤ 在 Panel2 中添加 ShellTreeView 组件 ShellTreeView1，设置 ShellTreeView1 的

Align 属性为 alClient。

⑥ 在 Panel3 中添加 ShellListView 组件 ShellListView1，设置 ShellListView1 的 Align 属性为 alClient。

调整好上述组件的大小和位置，如图 11-5 所示。

(2) 设置文件、文件夹管理类组件的属性，使 ShellComboBox 组件、ShellTreeView 组件和 ShellListView 组件相互关联。属性设置参见表 11-3、表 11-4 和表 11-5。

表 11-3 ShellComboBox 组件的属性设置

属性名称	属性值
ShellListView	ShellListView1
ShellTreeView	ShellTreeView1
Root	rfMyComputer

表 11-4 ShellTreeView 组件的属性设置

属性名称	属性值
ShellListView	ShellListView1
ShellComboBox	ShellComboBox1
Root	rfMyComputer

表 11-5 ShellListView 组件的属性设置

属性名称	属性值
ShellComboBox	ShellComboBox1
ShellTreeView	ShellTreeView1
Root	rfMyComputer

(3) 运行程序，结果如图 11-6 所示。

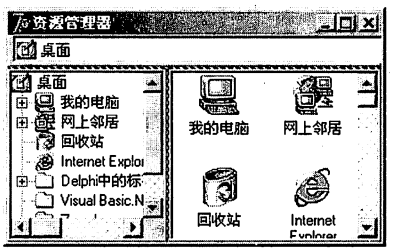


图 11-5 程序界面

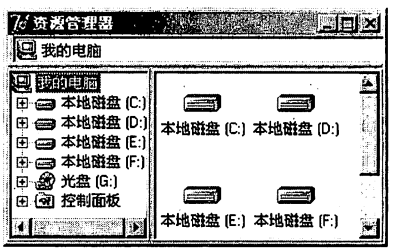


图 11-6 程序运行结果

11.2 文件管理的相关函数与过程

Delphi 提供了很多非常好的函数和过程，用户可以利用它们很好地管理 Windows 文件和文件夹。下面将具体介绍其中的部分函数和过程。

1. ReNameFile 函数

ReNameFile 函数用于改变文件名。函数原型为：

```
function RenameFile(const OldName, NewName: string): Boolean;
```

说明：

(1) OldName 是要修改的文件的文件名，NewName 是新文件名。

(2) 如果 NewName 文件已经存在，则改名失败，函数返回 False；否则，修改成功，返回 True。

(3) 新老文件可以不在一个文件夹，甚至可以不在一个磁盘分区。

例如：

```
if not Renamefile('c:\old.txt','d:\new.txt')
then showmessage('改名失败!');
```

2. DeleteFile 函数

DeleteFile 函数用于删除一个文件。函数的原型为：

```
function DeleteFile(const FileName: string): Boolean;
```

说明：

(1) FileName 为包含路径在内的文件名。

(2) 如果删除成功，函数返回 True；否则，返回 False。

例如：

```
if not DeleteFile('d:\new.txt')
then showmessage('删除失败!');
```

3. FileExists 函数

FileExists 函数用于判断指定的文件是否存在。函数原型为：

```
function FileExists(const FileName: string): Boolean;
```

说明：如果 FileName 指定的文件存在，则函数返回 True；否则，函数返回 False。

4. FileGetAttr 函数

FileGetAttr 函数用于得到一个文件的属性。函数原型为：

```
function FileGetAttr(const FileName: string): Integer;
```

说明：

(1) 文件的属性是指文件是只读文件、隐藏文件、系统文件、文档文件中的哪种。

(2) 返回值为 faHidden(值为 2)、faReadOnly(值为 1)、faSysFile(值为 4)、faArchive(值为 32)，分别表示隐藏文件、只读文件、系统文件、文档文件。

例如：

```
Attrs:=FileGetAttr('d:\shan1.jpg');
if (Attrs and faHidden<>0 )
then FileSetAttr('d:\shan1.jpg', Attrs-faHidden);
```

无论文件'd:\shan1.jpg'是否是隐藏文件，上面的代码都可以将文件'd:\shan1.jpg'设置为非隐藏文件，而文件的其他属性值不变(只读文件与否、系统文件与否、文档文件与否均保持不变)。

5. FileSetAttr 函数

FileSetAttr 函数用于设置文件的属性，函数原型为：

```
function FileSetAttr(const FileName: string; Attr: Integer): Integer;
```

说明：

(1) 如果操作成功则返回 0，否则返回错误的值(非 0)。

(2) Attr 可以是多个文件属性的组合。

例如：

```
Attrs:=FileGetAttr('d:\shan2.jpg');
FileSetAttr('d:\shan2.jpg',Attrs or faHidden);
```

上面的代码可以给文件'd:\shan2.jpg'添加一个隐藏属性。而 FileSetAttr('d:\shan2.jpg', faReadOnly or faHidden)可以设置文件'd:\shan2.jpg'为只读文件和隐藏文件。

6. DirectoryExists 函数

DirectoryExists 函数用于判断一个文件夹是否存在。函数原型为：

```
function DirectoryExists(const Directory: string): Boolean;
```

说明：如果文件夹存在，函数返回 True；否则函数返回 False。

7. CreateDir 函数

CreateDir 函数用于创建指定的文件夹。函数的格式为：

```
function CreateDir(const Dir:string):Boolean;
```

说明：如果所指定的新文件夹已经存在，则创建失败；否则创建成功。如果创建成功返回 True，否则返回 False。

8. RemoveDir 函数

RemoveDir 函数用于删除指定的空文件夹。函数的格式为：

```
function RemoveDir(const Dir: string): Boolean;
```

说明：

(1) 可以删除指定的空文件夹。

(2) 删除成功返回 True，否则返回 False。

11.3 文件操作

前面的章节讲述了简单的数据类型以及枚举型、子界型、集合型、数组型、记录型等构造类型。这些数据类型的变量在程序运行时，Delphi 自动为它们分配存储单元，但是当程序运行结束后，这些变量不再保存，因此用户无法恢复这些数据。例如，要想编写程序

处理一个班多人的数据信息，可以使用记录数组，但是记录数组有如下两个缺点。

(1) 每次都需要输入大量的数据。

(2) 程序运行结束后，所有的数据都没有保存。

为了保存数据，有一个较好的办法是将这些数据以文件的形式保存到磁盘上，这样即使程序停止运行，这些数据也仍然保存在磁盘上，下次运行时无须重新输入。

基于文件的重要性，本节将讲述文本文件和有类型文件的操作方法。

11.3.1　文件类型

根据文件中数据元素的数据类型，Delphi 中的文件可以分为 3 种：文本文件、有类型文件和无类型文件。本章将具体讲述文本文件的操作和有类型文件的操作。

(1) 文本文件

文本文件的元素是字符，每个元素占据一个字节，以回车换行符表示每行的结束。

(2) 有类型文件

有类型文件的元素类型可以是除文件之外的各种数据类型。每个元素所占据的字节数由元素类型决定。

(3) 无类型文件

无类型文件通常以字节为单位对文件中的二进制数据元素进行操作，而不管每个字节表示什么类型的数据元素。

11.3.2　适合于各种文件的操作

(1) 与外部文件建立联系

在 Delphi 中对外部文件进行操作前，需要将外部文件与一个文件类型的变量关联。AssignFile 过程用于实现外部文件与文件变量的关联，过程的原型是：

```
procedure AssignFile(var F; FileName: string);
```

说明：

① F 是一个文件变量，它不仅可以表示文本文件还可以表示其他类型的文件。

② 调用该过程以后，F 就与外部文件发生关联，直到中断该文件变量与外部文件之间的联系为止。

(2) 与外部文件中断关系

文件操作结束后还需要中断外部文件与相应的文件变量的联系。CloseFile 过程用于中断外部文件与文件变量的联系。它的原型为：

```
procedure CloseFile(var F);
```

(3) 以读方式打开文件

Delphi 通过 Reset 过程以读方式打开文件，过程的原型为：

```
procedure Reset(var F [: File; RecSize: Word ] );
```

说明：

① 通过 Reset 过程可以打开一个已经存在的文件(各种文件类型)。如果该文件是

文本文件，则自动为只读文件。如果该文件不存在，则会产生一个错误。

② 如果指定的文件已经打开，则先关闭再打开，设置当前文件的位置为文件开始处。

③ 调用过程之后，如果文件为空，则 Eof(F)值为 True，否则值为 False。

④ RecSize 是一个默认表达式，仅当 F 为无类型文件时才使用这个参数。该参数指定数据传输时，文件的大小取默认值 128B。

(4) 以写方式打开文件

Delphi 通过 ReWrite 过程以写方式打开文件，过程的原型为：

```
procedure Rewrite(var F: File [; Recsize: Word ] );
```

说明：

① 通过 ReWrite 过程可以创建一个新文件。如果该文件是文本文件，则自动为只读文件。

② 如果指定的文件已经存在，则删除该文件再创建同名文件；如果指定的文件已经打开则先关闭，然后再重新创建，并设置当前位置为文件开始处。

③ 调用 ReWrite，则 Eof(F)必为 True。

④ RecSize 是一个默认表达式，仅当 F 为无类型文件时才使用这个参数。该参数指定数据传输时，文件的大小取默认值 128B。

(5) 删除文件

Delphi 调用 Erase 删除外部文件，过程的原型为：

```
procedure Erase(var F);
```

说明：F 是任意文件类型，F 先关联到外部文件，文件删除之前先关闭。

11.3.3 文本文件的操作

文本文件是使用最多的文件类型，Delphi 提供了各种操作方式对文本文件进行操作。如果要操作文本文件，必须首先使用 AssignFile 过程将文本文件与文件变量发生关联，再使用 ReSet、ReWrite 或者 Append 方式打开文件，然后对文件进行读写等操作。文本文件的读写只能够顺序进行，如果要读文件则必须将文件以读方式打开，如果要写文件则必须将文件以写方式打开。文件的读和写不能同时进行。

下面的操作只对文本文件有效。

(1) 以添加方式打开文件

Delphi 通过调用过程 Append 可打开一个已经存在的文件，并准备在文件末尾添加文本。如果该文件不存在，则会产生一个错误。过程的原型为：

```
procedure Append(var F:Text);
```

(2) 用 Read 过程读文件

Delphi 中的 Read 过程可以从文本文件中读取字符串、字符或者数字。过程的原型为：

```
procedure Read( [ var F:Text;]V1[,V2,…,Vn] );
```

其中，F 是文本文件变量，V1、V2、…、Vn 是用于存储读取的数据，它们类型相同，一般为字符型、字符串型、整型、实型。

(3) 用 ReadLn 过程读文件

读取以回车换行符结束的一行字符串，然后跳过回车符，再读下一行。过程原型为：

```
procedure ReadLn([var F:Text;]V1[,V2,…,Vn]);
```

其中，F 为文本文件变量，V1、V2、…、Vn 是用于存储读取的数据，它们类型相同，一般为字符型、字符串型、整型、实型。

(4) 用 Write 过程写文件

Delphi 通过调用 Write 过程向文件中写入数据。过程原型为：

```
procedure Write([varF:Text;]P1[,P2,…,Pn]);
```

其中，F 是文本文件变量，P1、P2、…、Pn 是要写入的数据，可以是字符型、字符串型、整型、实型变量。

(5) 用 WriteLn 过程写文件

Delphi 通过调用 WriteLn 过程向文件中写入数据。过程原型为：

```
procedure WriteLn([varF:Text;]P1[,P2,…,Pn] );
```

其中，F 是文本文件变量，P1、P2、…、Pn 是要写入的数据，可以是字符型、字符串型、整型、实型变量。

(6) Eoln 函数

该函数判断文件指针是否在行结尾。原型为：

```
function Eoln[(varF:Text)]:Boolean;
```

其中，F 是文本文件变量。

(7) SeekEof 函数

该函数可以返回文件尾状态。原型为：

```
function SeekEof[(varF:Text)]:Boolean;
```

其中，F 是文本文件变量。如果文件位置在文件末尾则返回 True，否则返回 False。

(8) SeekEoln 函数

该函数可以返回文件的行尾状态，原型为：

```
function SeekEoln[(varF:Text)]:Boolean;
```

其中，F 是文本文件变量。如果文件位置在行结尾则返回 True，否则返回 False。

【例 11-3】 编写程序，完成如图 11-7 的文件读写工作。

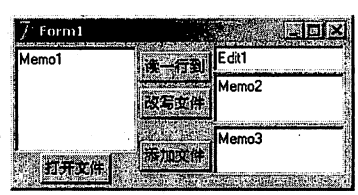


图 11-7　程序界面

设计步骤如下：

(1) 添加组件，在窗体上添加组件 Memo1、Memo2、

Memo3、Edit1、Button1(打开文件)、Button2(读一行到)、Button3(改写文件)和 Button4(添加文件)。

(2) 编写程序如下。

① 编写“打开文件”的代码如下:

```
var f:textfile;                          //文件变量

procedure TForm1.Button1Click(Sender: TObject);

 var line:string;

begin
 memo1.Text:='';
 assignfile(f,'静夜思.txt');
 reset(f);
 while not eof(f) do                     //判断文件是否结尾
 begin
  readln(f,line);
  memo1.Lines.Add(line);                 //读到的每一行显示在多行编辑框 Memo1
 end;
 closefile(f);
end;
```

说明:打开文件实际上就是读文件'静夜思.txt'的全部内容到多行编辑框 Memo1,因此可以使用循环的方法一直读到文件结尾。程序运行结果如图 11-8 所示。

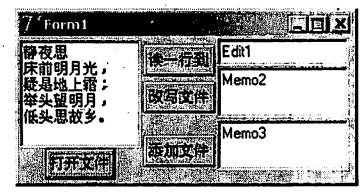


图 11-8 打开文件

② 编写“读一行到”的代码如下:

```
procedure TForm1.Button2Click(Sender:TObject);

 var line:string;

begin
 memo1.Text:='';
 assignfile(f,'静夜思.txt');
 reset(f);
 readln(f,line);
 edit1.Text:=line;
 closefile(f);
end;
```

说明:从文件'静夜思.txt'的最前面读一行到单行编辑框 Edit1。

③ 编写“改写文件”的代码如下:

```
procedure TForm1.Button3Click(Sender: TObject);

 var line:string;i,lastline:integer;
```

```
begin
 lastline:=memo2.Lines.Count;
 assignfile(f,'静夜思.txt');
 rewrite(f);
 for i:=0 to lastline-1 do
 writeln(f,memo2.lines[i]);
 closefile(f);
end;
```

说明：本程序中改写文件，实际上就是用多行编辑框 Memo2 中的内容替代原有的文件。程序运行结果如图 11-9 所示。

④ 编写“添加文件”的代码如下：

```
procedure TForm1.Button4Click(Sender: TObject);

 var line:string;i,lastline:integer;

begin
 lastline:=memo3.Lines.Count;
 assignfile(f,'静夜思.txt');
 append(f);
 for i:=0 to lastline-1 do
 writeln(f,memo3.lines[i]);
 closefile(f);
end;
```

说明：添加文件实际上就是在原来的文件之后添加新的内容，此处添加了两行内容，程序运行结果如图 11-10 所示。

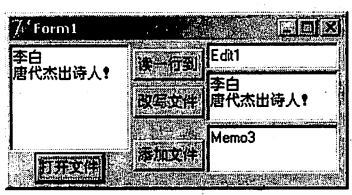


图 11-9　改写文件(原文件完全改变)

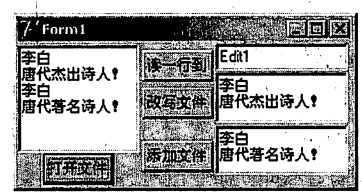


图 11-10　添加文件(在③的基础上添加)

11.3.4　有类型文件的操作

有类型文件是指文件的元素具有一定的数据类型，而读写过程所操作的元素是指定类型的数据。有类型文件的声明格式如下：

```
var<有类型文件变量>:file of<类型>;
```

其中，＜有类型文件变量＞是用户所定义的文件变量，＜类型＞是指文件元素的数据类型。由于文件元素的类型已指定，每条记录的长度是固定的，因此用户可以随机访问各条记录。

有类型文件的操作方式如下：

(1) 有类型文件的读操作。对于有类型的文件，Delphi 通过过程 Read 完成读操作，

Read 过程的原型为：

```
procedure Read(F,V1[,V2,…,Vn] );
```

其中，F 是有类型文件的文件变量，V1、V2、…、Vn 是类型变量，读取的数据分别赋值给这些类型变量，并且将文件指针向后移动。

(2) 有类型文件的写操作。Delphi 通过过程 Write 实现有类型文件的写入操作，Write 过程的原型为：

```
procedure Write(F,V1,…,Vn);
```

其中，F 是类型文件的文件变量，V1、V2、…、Vn 是要写入的数据，类型与文件中数据的类型一致。

(3) FilePos 函数可以返回文件的位置，原型为：

```
function FilePos(var F):Longint;
```

其中，F 是有类型文件的文件变量。返回值为文件位置。

(4) FileSize 函数可以求出文件的大小，原型为：

```
function FileSize(var F): Integer;
```

其中，F 是有类型文件的文件变量，其返回值是文件的长度。

(5) Seek 过程用于将文件指针指向文件位置，过程原型为：

```
procedure Seek(var F; N: Longint);
```

【例 11-4】 编写程序，对班上的学生信息进行管理，学生信息包括学号、姓名、语文、数学。要求该程序具有信息录入、浏览、查询、添加和删除功能。

设计步骤如下：

(1) 添加组件。添加标签 4 个、单行编辑框 4 个、命令按钮 7 个(从左至右、从上至下依次为 Button1～Button7)。添加列表框 1 个、面板组件 1 个和组框组件 1 个。进行属性设置(省略)，界面如图 11-11 所示。

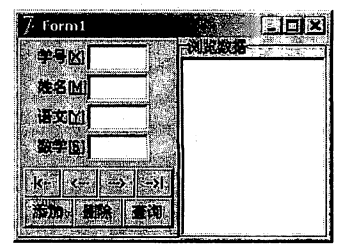


图 11-11 程序界面

(2) 编写代码如下。

① 先定义类型和全程变量：

```
type
 studata=record
  xh:string[2];
  xm:string[6];
  yw,sx:integer;
 end;

var
 recno:integer;
```

说明：recno 表示当前记录位置，要在多处使用并修改它，因此要定义为全程变量。

② 编写 Form1 的 OnCreate 事件过程：

```
procedure TForm1.FormCreate(Sender: TObject);
 var
  t:studata;f:file of studata;

begin
 assignfile(f,'stu.txt');
 if fileexists('stu.txt') then
   reset(f)
 else
  rewrite(f);
 while not eof(f)do
 begin
  read(f,t);                          //读文件所有内容显示到列表框中
  listbox1.Items.Add(t.xh+'  '+t.xm+'  '+inttostr(t.yw)+'  '+inttostr(t.sx));
 end;
 if filesize(f)> 0 then          //如果文件存在且不为空,则读第 1 条记录显示在编辑框中
 begin
  recno:=0;
  seek(f,recno);                      //指向第 1 条记录
  read(f,t);                          //读第 1 条记录
  edit1.Text:=t.xh;
  edit2.Text:=t.xm;
  edit3.Text:=inttostr(t.yw);
  edit4.Text:=inttostr(t.sx);
  listbox1.ItemIndex:=recno;          //列表框也指向第 1 条记录
 end;
 closefile(f);
end;
```

说明：本过程在程序运行之初执行，它判断文件 stu.txt 是否存在。如果存在，则在列表框中显示文件，并置文件位置为第 1 条记录，将第 1 条记录显示在编辑框中。如果文件不存在，则建立该文件。

③ 编写“添加”(即“刷新”)按钮代码：

```
procedure TForm1.Button5Click(Sender: TObject);
 var t:studata;
     f:file of studata;
     size:integer;

begin
 if Button5.Caption='添加' then
 begin
  edit1.Text:='';                 //置空编辑框为输入做好准备
  edit2.Text:='';
  edit3.Text:='';
  edit4.Text:='';
  edit1.SetFocus;
  Button5.Caption:='刷新'
```

```
 end
 else
 begin                          //将输入的内容保存到文件中
  t.xh:=edit1.Text;
  t.xm:=edit2.Text;
  t.yw:=strtoint(edit3.Text);
  t.sx:=strtoint(edit4.Text);
  assignfile(f,'stu.txt');
  reset(f);
  size:=filesize(f);
  seek(f,size);                 //文件指针置于文件末尾
  write(f,t);                   //将 t 变量写入文件末尾
  listbox1.Items.Clear;         //清空列表框
  seek(f,0);                    //定位第 1 条记录,将文件所有记录重新读出并显示在列表框中
  while not eof(f) do
  begin
   read(f,t);
   listbox1.Items.Add(t.xh+'  '+t.xm+'  '+inttostr(t.yw)+'  '+inttostr(t.sx));
  end;
  closefile(f);
  edit1.Text:='';
  edit2.Text:='';
  edit3.Text:='';
  edit4.Text:='';
  edit1.SetFocus;
  Button5.Caption:='添加'
 end;
end;
```

说明：本过程在文件最后添加记录。按“添加”按钮，编辑框置空，“添加”按钮变成“刷新”按钮。在编辑框中输入数据后，按“刷新”按钮完成记录的保存，“刷新”按钮变成“添加”按钮。

④ 编写[illegible]的代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);

 var t:studata;
     f:file of studata;

begin
 assignfile(f,'stu.txt');
 reset(f);
 if filesize(f)>0 then                  //如果文件不为空
 begin
  recno:=0;
  seek(f,recno);                        //指向第 1 条记录
  read(f,t);                            //读第 1 条记录
  edit1.Text:=t.xh;
  edit2.Text:=t.xm;
```

```
  edit3.Text:=inttostr(t.yw);
  edit4.Text:=inttostr(t.sx);
  listbox1.ItemIndex:=0;                    //列表框也指向第 1 条记录
 end;
 closefile(f);
end;
```

说明：本过程读第 1 条记录显示在编辑框中。

⑤ 编写<-的代码如下：

```
procedure TForm1.Button2Click(Sender: TObject);

 var t:studata;
     f:file of studata;

begin
 assignfile(f,'stu.txt');
 reset(f);
 if filesize(f)>0 then                      //如果文件不为空
 begin
  recno:=recno-1;
  if recno<0 then recno:=0;
  seek(f,recno);                            //指向前一条记录
  read(f,t);
  edit1.Text:=t.xh;
  edit2.Text:=t.xm;
  edit3.Text:=inttostr(t.yw);
  edit4.Text:=inttostr(t.sx);
  listbox1.ItemIndex:=recno;
 end;
 closefile(f);
end;
```

说明：该过程显示前一条记录。

⑥ 编写->的代码如下：

```
procedure TForm1.Button3Click(Sender: TObject);

 var t:studata;
     f:file of studata;

begin
 assignfile(f,'stu.txt');
 reset(f);
 if filesize(f)>0 then                      //如果文件不为空
 begin
  recno:=recno+1;
  if recno> =filesize(f) then recno:=filesize(f)-1;
  seek(f,recno);                            //文件指向后面一条记录
```

```
  read(f,t);
  edit1.Text:=t.xh;
  edit2.Text:=t.xm;
  edit3.Text:=inttostr(t.yw);
  edit4.Text:=inttostr(t.sx);
  listbox1.ItemIndex:=recno;
 end;
 closefile(f);
end;
```

说明：该过程显示后一条记录。

⑦ 编写->|的代码如下：

```
procedure TForm1.Button4Click(Sender: TObject);

var t:studata;
     f:file of studata;

begin
 assignfile(f,'stu.txt');
 reset(f);
 if filesize(f)>0 then
 begin
  seek(f,filesize(f)-1);
  recno:=filesize(f)-1;
  read(f,t);
  edit1.Text:=t.xh;
  edit2.Text:=t.xm;
  edit3.Text:=inttostr(t.yw);
  edit4.Text:=inttostr(t.sx);
  listbox1.ItemIndex:=recno;
 end;
 closefile(f);
end;
```

说明：该过程显示最后一条记录。

⑧ 编写 ListBox1 的 OnClick 事件过程如下：

```
procedure TForm1.ListBox1Click(Sender: TObject);

var t:studata;
     f:file of studata;

begin
 assignfile(f,'stu.txt');
 reset(f);
 if listbox1.ItemIndex>=0 then
  begin
   recno:=listbox1.ItemIndex;
   seek(f,recno);       //文件指向第 recno+1 条记录
```

```
    read(f,t);
    edit1.Text:=t.xh;
    edit2.Text:=t.xm;
    edit3.Text:=inttostr(t.yw);
    edit4.Text:=inttostr(t.sx);
    listbox1.ItemIndex:=recno;
  end;
 closefile(f);
end;
```

说明：该过程的作用是单击列表框可以改变当前记录位置。例如，单击第 n 条记录，则在编辑框中显示第 n 条记录。

备注：请读者自己完成删除和查询的代码。

(3) 运行程序，如图 11-12 所示。

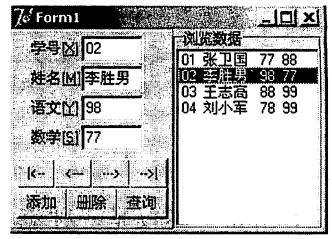


图 11-12　程序运行结果

11.4　小结

本章讲述了与文件有关的组件、函数和过程，还讲述了文件的操作。文件操作包含文件的读写、存储、文件指针操作等，还包含文本文件操作、有类型文件操作和无类型文件操作。

通过本章的学习，读者应该做到：

- 掌握文件管理组件的使用，特别是 FileList 组件的使用，以及如何让这些组件协调工作。
- 掌握与文件操作有关的函数与过程，特别是与读写操作有关的函数与过程。
- 文件操作，特别是文本文件的操作和有类型文件的操作。

习题

1. 编写一个简易的编辑器，该编辑器可以新建、保存、打开文件。要求使用文本文件的相关知识。

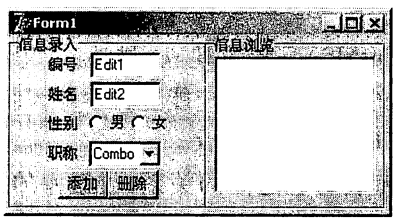


图 11-13　程序界面

2. 输出杨辉三角形到文本文件。

3. 编写程序，完成计算机系教师信息的输入、浏览、查询、删除。界面如图 11-13 所示。其中，教师信息包括编号、姓名、性别、职称(职称包括助教、讲师、副教授和教授)。

4. 完成例 11-4 中的删除、查询功能。

5. 将例 11-4 中语文或数学不及格的学生记录保存为一个新文件 stubjg. dat。

6. 将例 11-4 中平均分在 85 分以上的学生姓名保存为一个新文件 stuname. dat。

第 12 章

数据库编程基础

Delphi 7 不仅是一种流行的程序设计语言，同时也是一种功能强大的数据库应用程序开发工具。利用 Delphi 7 可以开发出各种数据库应用程序，其强大的数据库功能具体表现之一就是它能够访问和操作多种类型的数据源。在开发数据库时，用户可以通过 BDE(Borland Database Engnine)或者 ADO(Active Data Object)来访问各种数据源。另外，Delphi 还提供了丰富的工具，例如 Database Desktop 和 BDE Administrator，用户可以通过这些工具很方便地操作数据库。

12.1 数据库的基本概念

12.1.1 数据和数据库

计算机要处理数据，必须首先将数据按照一定的格式保存到计算机中。这里所说的数据，可以是数字、符号、图像、声音等。所谓数据库(Database，DB)，就是保存数据的仓库，其特点是数据具有一定的格式，数据高度结构化，可以供多个用户共享，且具有一定的安全性。

数据库分为关系型数据库、层次型数据库和网状型数据库，其中关系型数据库使用最为广泛。关系型数据库按照二维表格的形式组织存放数据，二维表的列称为字段，行称为记录。本书所讲述的数据库都是关系型数据库。

12.1.2 数据库管理系统(DBMS)

数据库管理系统(Database Management System，DBMS)是提供建立、描述、管理和维护数据库的程序系统，是数据库系统的核心部分。它的目的是使用户能够科学地组织、存储、访问、管理、维护数据。数据库管理系统主要有如下功能。

(1) 数据定义功能：包括数据库的结构定义、数据库表与表之间的关系，数据库的控制权限、口令、数据字典等。

(2) 数据操作功能：包括数据的检索、插入、修改、删除等基本操作。用户可以通过语言、组件等手段操作数据库中的数据。

(3) 数据库的建立与维护：包括数据库的结构定义、数据输入、数据修改、数据维护、数据更新等。

(4) 数据通信功能：使用 Delphi 可以很方便地编写出多层分布式数据库应用系统或者基于网络的数据库应用系统。通过这些数据库应用系统可以非常方便地实现计算机终端与终端之间、多个计算机终端之间、多台计算机之间的通信。

12.1.3　关系数据库

关系数据库(Relational Database)是由若干张表构成的，表与表之间存在密切的联系。在 Delphi 中，数据库的概念和其他数据库系统中的数据库概念有些不同，它指的是存放表的目录。

表(Table)是一些数据按照某种格式存放形成的文件，是二维的，行表示记录，列表示字段。某成绩表 cj. db 如图 12-1 所示。

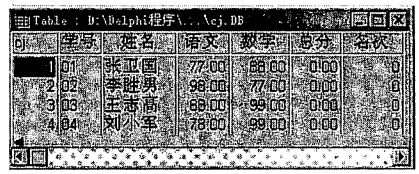


Table : D:\Delphi程序\...\cj.DB

	学号	姓名	语文	数学	总分	名次
1	01	张卫国	77.00	88.00	0.00	0
2	02	李胜男	98.00	77.00	0.00	0
3	03	王志高	69.00	99.00	0.00	0
4	04	刘小军	78.00	99.00	0.00	0

图 12-1　成绩表 cj. db

从图 12-1 可知有如下特性。

(1) 字段(Field)：表中的每个列称之为字段，例如学号、姓名就是字段，每个字段都有数据类型和宽度，例如学号的类型是 Alpha(字符串类型)，数据宽度为 2。

(2) 索引(Index)：索引可以使查询更加快捷，一个表可以建立多个索引。

(3) 关键字(Key Word)：关键字可以用来唯一确定某条记录，例如学号是唯一的，学号为“01”的记录就只有“张卫国”这一条。

(4) 记录(Record)：表中的每一行就是一条记录，例如学号为“02”的学生信息就是一条记录。

12.2　数据库辅助工具

Delphi 提供了可使开发 Delphi 数据库应用程序更加方便的辅助工具。

Database Desktop(数据库工作平台)：用户可以使用它创建数据库表，查询、修改数据库表结构，修改数据库表记录等。

BDE Administrator(数据库引擎管理器)：利用它可以建立数据库别名，实现数据库应用程序与数据库之间的连接。

ODBC(开放式数据库链路)：使 Delphi 程序可以访问 Delphi 不支持的数据库。

SQL Explorer(数据库管理器)：主要用来浏览数据库。

12.2.1　Database Desktop

Database Desktop 是 Delphi 最重要的工具，利用它用户可以建立各种类型的数据库表、查询、修改表结构，修改表记录等。选择 Windows“开始”菜单，选择“程序”|Borland Delphi 7|Database Desktop，出现 Database Desktop 界面，如图 12-2 所示。如果用户已经进入 Delphi 7，可以选择菜单 Tool|Database Desktop 进入 Database Desktop 界面，这样可能会更加快捷。

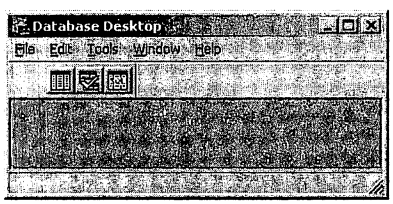


图 12-2　Database Desktop 界面

1. 设置工作目录

Database Desktop 具有两个目录：工作目录和私有目录。工作目录是数据库工作平台首先要找文件的地方，私有目录是用户自己的目录，其他网络用户无法看到。设置工作目录在 File 菜单中进行。

其中 Working Directory... 用于设置数据库所在的路径(即工作目录)，Alias 用于指明数据库别名。Private Directory... 用于设置私有目录，用户可以直接在文本框中输入私有目录或者单击 Browse... 按钮选择某个目录作为私有目录。工作目录和私有目录的设置如图 12-3 所示。

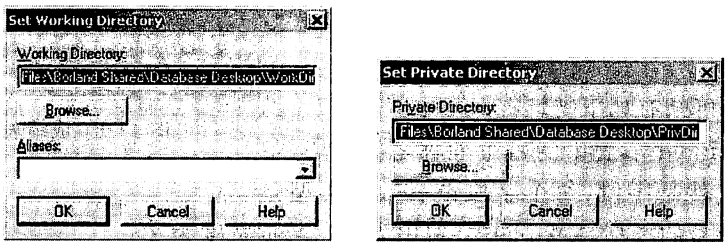


图 12-3 设置工作目录和私有目录

2. 创建数据库表

使用 Database Desktop 可以创建桌面的数据库表，如 Paradox、Access 等。下面以创建图 12-1 所示的成绩表 cj. db 为例来讲述使用 Database Desktop 建表的步骤。

(1) 执行菜单命令 File|New|Table，弹出 Create Table 对话框，如图 12-4 所示，在对话框中选择数据库的类型为默认的 Paradox 7，单击 OK 按钮，弹出建表对话框，如图 12-5 所示。

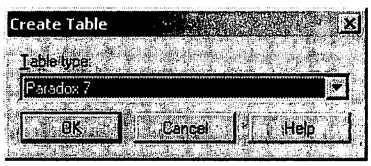


图 12-4 Create Table 对话框

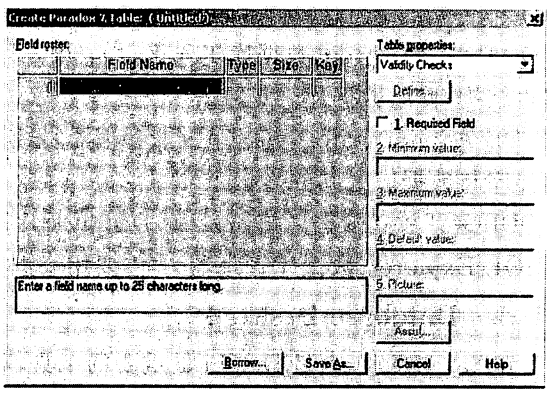


图 12-5 建表对话框

在 Create Paradox 7 Table 对话框中建立成绩表 cj. db。

(2) 定义表结构

定义第一个字段学号，在 Field Name 中输入第一个字段的名称“学号”，类型(Type)定义为 Alpha 类型(字符串类型)，鼠标右键单击 Type 下面的空白处即弹出类型供用户

选择。宽度(Size)定义为 2。用鼠标双击 Key 下面的空白处可以为学号添加主键(关键字),因为班上所有学生的学号都是独一无二的,每个学生的学号不允许相同。添加主键后,该列出现一个 * 号,表示主键设置成功。

表 12-1 给出了 Paradox 7 表的数据类型。

表 12-1　Paradox 7 表的数据类型及含义

符号	名　称	说　明	符号	名　称	说　明
A	Alpha	字符类型,长度 1～255	N	数值类型	只能输入数值类型
$	Money	货币类型	S	Short	范围－32767～32768
I	Long Integer	长整数	#	BDC	范围 0～32
D	Date	日期类型,如 2006-9-5	T	Time	时间,如 12:12:38
M	Memo	长度 1～240	F	Formatted Memo	设置输入文本的格式
G	Graphic	长度 0～240,显示图像	O	Ole	Ole 连接
L	Logical	逻辑值 True 或者 False	+	Autoincrement	自动增 1
B	Binary	0～32,用于声音文件等			

按照同样的方法定义其他字段。成绩表 cj. db 定义后的结构如图 12-6 所示。

其中,为总分定义了一个索引,该索引是降序索引,如图 12-7 所示。

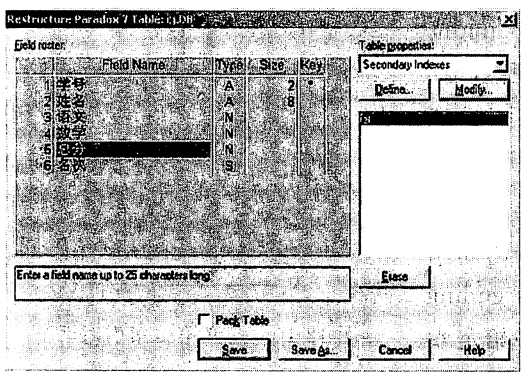


图 12-6　成绩表 cj. db 的表结构

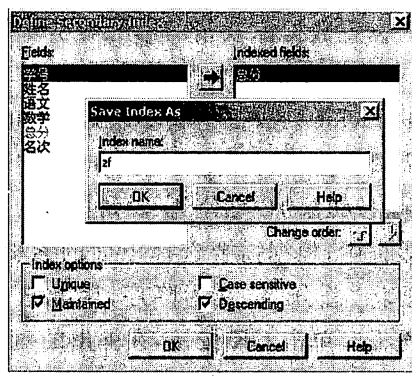


图 12-7　为总分定义索引

结构定义完毕,单击 Save As 按钮保存文件为 cj. db,路径为“d:\delphi 程序\ch12\eg1”。

(3) 输入记录

在 Database Desktop 中,单击 File | Open | Table 菜单,打开文件 cj. db,出现如图 12-8 所示的界面。

选择菜单 Table | Edit Data,此时可以输入或编辑数据记录。输入记录后的表如图 12-9 所示。如果想再次修改表结构,可以选择 Table | Edit Data | Restructure,然后再修改表结构,结构修改完毕,按 Save 按钮保存或者按 Save As 按钮重新命名。

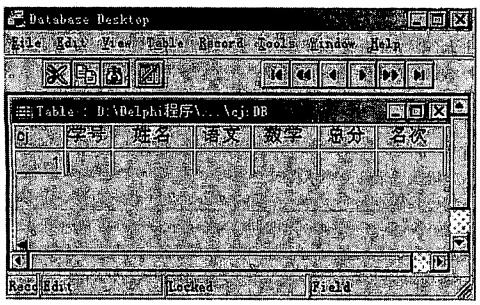


图 12-8 记录输入界面

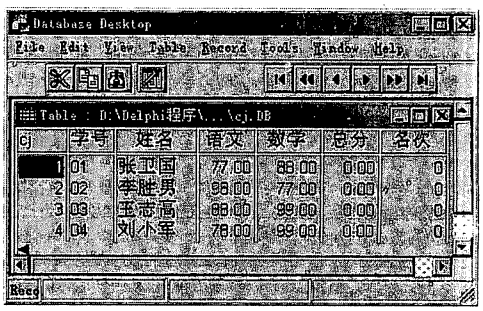


图 12-9 输入记录后的表

3. 数据库表的属性

细心的读者一定会发现，图 12-6 表结构界面的右边 Table Properties 下拉列表框中提供了一个属性列表，在这里用户可以对数据库表的各种属性进行必要的设置。数据库表的主要属性有有效性检查、第二索引、口令等。

(1) Validity Checks

选择 Required Field 前的复选框，表示该字段不能为空；Mininum Value 表示输入字段的最小值；Maxinum Value 表示输入的字段的最大值；Default Value 表示输入字段的默认值。Picture 表示字段的显示格式。

(2) Secondary Index

设置数据库表格的第二索引。单击 Define 按钮，弹出定义第二索引对话框，如图 12-10 所示。左边的 Fields 列表框中就是数据库表的所有字段，选择需要定义第二索引的字段，例如“总分”字段，单击 按钮添加“总分”字段到右边的 Indexed fields 列表框中，根据需要选择下面的索引选项，例如用户要将“总分”定义为降序，因此选定 Descending 复选框。单击 OK 按钮，在弹出的对话框中给定索引的名称，例如“zf”，此时就为该字段定义了第二索引。

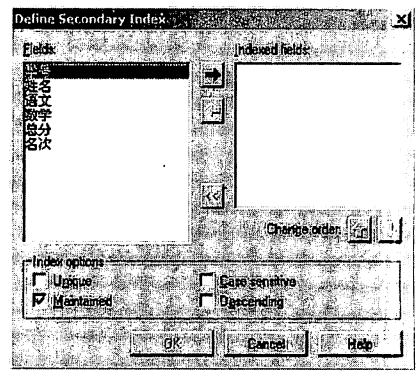


图 12-10 定义第二索引对话框

12.2.2 BDE Administrator

BDE Administrator(数据库引擎管理器)是 BDE 的配置程序，它主要用来对数据库的应用别名进行管理，配置驱动程序信息。在 Delphi 中组件要访问数据库表，必须先给定数据库表的位置(别名)。运行 BDE Administrator 有两种方式：打开 Windows 的控制面板，双击 BDE Administrator 图标；或者在“程序”菜单中选择 Borland Delphi 7 | BDE Administrator 。打开的 BDE Administrator 界面如图 12-11 所示。

BDE Administrator 管理器包含左边和右边两个部分。左边部分有两个选项卡，其中 Databases 选项卡中列出的是数据库别名，用户还可以添加新的。而 Configuration 选

项卡中显示的是数据库的配置参数，用户可以在这里修改数据库的参数。右边部分是用于显示当前选项的配置情况，当用户在左边选择某个选项的时候，右边将会显示该选项的配置信息。

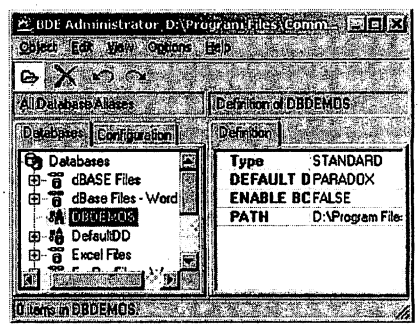


图 12-11　BDE Administrator 界面

下面介绍 BDE Administrator 的操作。

(1) 查看 BDE 的数据库别名定义。

单击数据库页标签可以显示出别名列表，选择一个别名后，就可以在右边 Definition 中查看它的有关属性和定义。

(2) 修改 BDE 数据库别名定义。

先在左边选择一个要修改的数据库别名，然后在右边 Definition 中设置各种属性，修改完成后单击工具栏上的“提交”按钮即可。或者鼠标右键单击数据库别名选择菜单 Apply 也可。

(3) 建立新的数据库别名。

下面以为 cj.db 建立一个数据库别名为例讲述数据库别名的建立方法。在 Database 页中单击鼠标右键，在弹出的菜单(如图 12-12 所示)中选择 New，显示如图 12-13 所示的对话框。选择 Standard，然后单击 OK 按钮，此时在左边多出一个数据库别名 Standard1，将其改名为“成绩管理”，此时在右边 Difinition 页中就会显示数据库别名为“成绩管理”的一些信息，如图 12-14 所示。

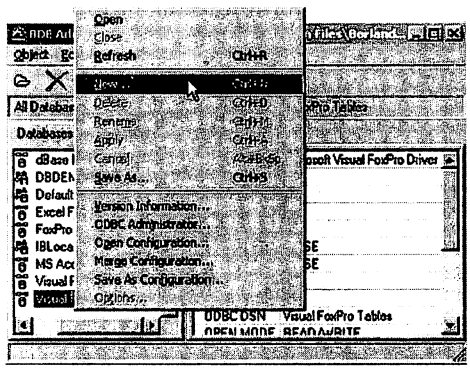


图 12-12　新建数据库别名

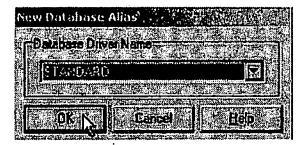


图 12-13　新建数据库别名对话框

若对“成绩管理”的属性进行设置，则只需要设置 PATH 属性即可。单击 PATH 后面的按钮，设置“成绩管理”的 PATH 为表 cj.db 所在的路径“D:\Delphi 程序\ch12\eg12-1”，如图 12-15 所示。

需要说明的是，不同数据库类型的右边 Definition 中显示的属性栏目也不一样。

(4) 删除数据库别名

以删除数据库别名“成绩管理”为例。鼠标右键单击数据库别名“成绩管理”，在弹出的菜单中选择 Delete 即可。

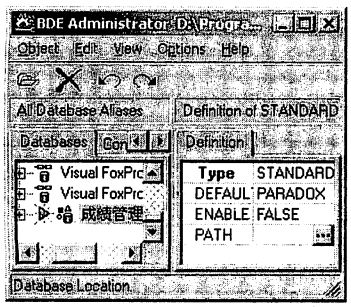


图 12-14 新建的“成绩管理”数据库别名

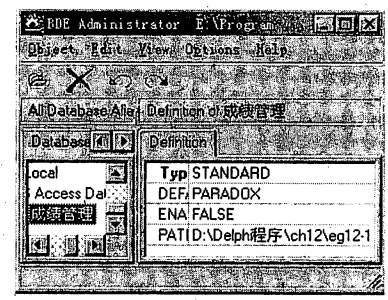


图 12-15 设置“成绩管理”的 PATH

12.2.3 SQL 资源管理器(SQL Explorer)

(1) 浏览数据库与建立数据库。利用 SQL 资源管理器浏览数据库与建立数据库的操作方式与利用 BDE Administrator 的操作方式类似。请读者参考 12.2.2 小节的相关知识点。

(2) 修改数据库与删除数据库。操作方法与 BDE Administrator 的操作方法类似。请读者参考 12.2.2 小节的相关知识点。

(3) 加密。双击 Database 选项卡中的一个数据库,将弹出 Database Login 对话框,如图 12-16 所示。用户输入用户名和口令,并单击 OK 按钮进行加密。

(4) 访问数据库的内容。选择数据库的一个表,例如 DBDEMOS 中的表 animals.dbf,此时工具栏中出现导航器,右边出现 3 个选项卡:Definition、Data 和 Enter SQL(如图 12-17 所示)。

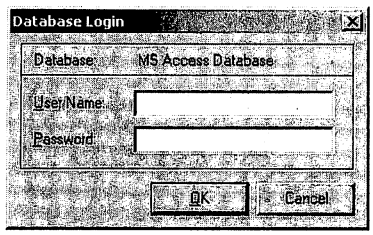


图 12-16 Database Login 对话框

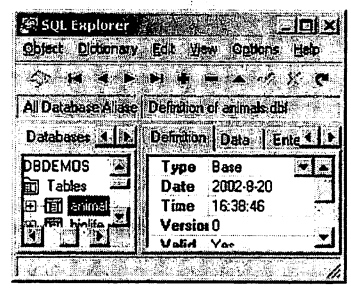


图 12-17 SQL Explorer

如果要定义表参数,请选择 Definition 选项卡。

如果要浏览数据库表记录,则选择 Data 选项卡,可以通过工具栏中的导航器进行浏览、编辑,如图 12-18 所示。

如果要执行 SQL 查询语句,则选择 Enter SQL 选项卡,并输入相应的 SQL 语句。如果要查询 country.db 中所有国家,则可以在 Enter SQL 文本框中输入 select from country 并按 Execute Query 按钮,显示信息如图 12-19 所示。

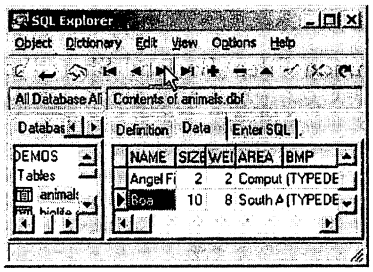


图 12-18　SQL Explorer Data 选项卡

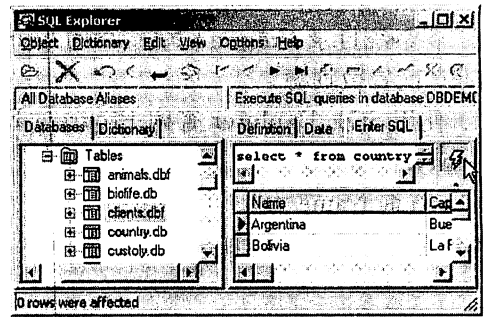


图 12-19　在 Enter SQL 选项卡中执行查询

12.3　小结

本章讲述了数据库的基本概念，还介绍了数据库的辅助工具，例如 Database Desktop、BDE Administrator 和 SQL 资源管理器。

通过本章的学习，读者应该掌握如下内容。

- 数据库的基本概念；
- 数据库工具 Database Desktop；
- 数据库工具 BDE Administrator；
- SQL 资源管理器。

习题

1. 使用 Database Desktop 建立一个数据库，包含 2 张 Paradox 表，其中 student 表用来描述学生信息（如表 12-2 所示），course 表用来描述课程信息（如表 12-3 所示）。

表 12-2　student. db 表结构

Field name	type	size	key	说　明
stu_no	a	2	*	学号
stu_name	a	15		姓名
stu_birthday	d			出生日期
stu_class	a	15		所在班级

表 12-3　course. db 表结构

fieldname	type	size	key	说　明
class_no	a	4	*	课程号
class_name	a	15		课程名
class_before	a	4		前导课
class_score	s			学分

2. 利用 BDE Administrator 设置第 1 题中的数据库别名为 stud。

3. 将系统目录 Program Files 下的 Common Files 中的 Borland Shared 中的 data 使用 ODBC 配置数据源。

4. 使用 SQL Explorer 查询 Program Files\Common Files\Borland Shared\data 中的 country. db 表。

(1) 查询 Name 为“chile”的记录。

(2) 查询 continent 为“South America”的记录。

第 13 章

BDE 数据库应用程序开发

Delphi 要访问数据库必须首先建立数据库应用程序和数据库之间的联系，Delphi 7 的数据访问组件和 BDE 组件就提供了这种联系方法。在 Delphi 中，开发数据库应用程序大都通过数据库的组件来实现。利用 BDE 组件开发数据库应用程序一般涉及三类组件：BDE 组件、Data Access 组件和 Data Controls 组件。这些组件的关系如图 13-1 所示。

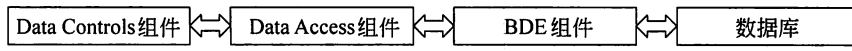


图 13-1　利用 BDE 组件访问数据库

BDE 组件位于 BDE 组件面板中，其作用是直接连接到物理数据库。它能直接读取数据库信息或者把修改的信息写入数据库，主要有 Ttable 组件、Tquery 组件等。BDE 组件又叫做数据集组件，如图 13-2 所示。

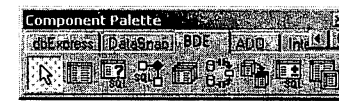


图 13-2　BDE 组件

Data Access 组件位于 Data Access 组件面板中，该类组件负责为 Data Controls 组件提供数据源，可直接与 BDE 组件发生联系，通过 BDE 组件访问数据库文件。Data Access 组件如图 13-3 所示。

Data Controls 组件位于 Data Controls 组件页中，该类组件负责显示数据库中的数据，是可视化组件，常用组件有：TDBNavigator 组件、TDBEdit 组件、TDBGrid 组件、TDBMemo 组件等。Data Controls 组件如图 13-4 所示。

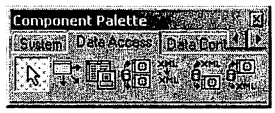


图 13-3　Data Access 组件

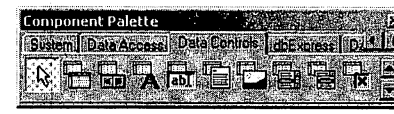


图 13-4　Data Controls 组件

13.1　Table 组件

Table 组件是开发数据库应用程序的最常见组件，它用于连接数据库的表。通过Table组件，用户可以对表进行存储访问、编辑修改等各种操作。Table组件提供了非常

丰富的属性、方法和事件，使用户开发数据库应用程序变得非常方便和轻松。

13.1.1 Table 组件的常用属性

1. DatabaseName 属性

DatabaseName 属性用来指定数据库名称，一般是别名。用户可以在属性页中设置该属性，也可以在 Form 的 OnCreate 事件过程中设置该属性。

假设在窗体上添加了组件 Table1，Table1 要访问在第 12 章中建立的数据库“成绩管理”，Table1 的 DatabaseName 可以设置为：Table1.DatabaseName := '成绩管理'，当然也可以使用路径，因此上面的语句可以写成 table1.Databasename :='d:\delphi 程序\ch12\eg12-1'。为了程序的可移植性，可以将数据库和应用程序放在同一个目录下（或者将数据库放在应用程序目录的子目录中），然后使用相对路径。例如：

```
procedure TForm1.FormCreate(Sender: TObject);

 var curdir:string;

begin
 //...
 getdir(0,curdir);                          //取得应用程序的目录,保存到 curdir 中
 table1.Databasename:=curdir;               //数据库别名就是应用程序的目录
 //...
end;
```

2. TableName 属性

TableName 属性用来为 Table 组件指定一个操作数据库表文件，与 DatabaseName 配合使用，应先设置 DatabaseName 属性后再设置 TableName 属性。正确设置 DatabaseName 属性之后，可以在 TableName 组合框中选择一个已经存在的表文件，如图 13-5所示。也可以使用代码来设置。如：

```
procedure TForm1.FormCreate(Sender: TObject);

 var curdir:string;

begin
 //...
 getdir(0,curdir);                          //getdir 可以得到程序所在的目录
 table1.Databasename:=curdir;
```

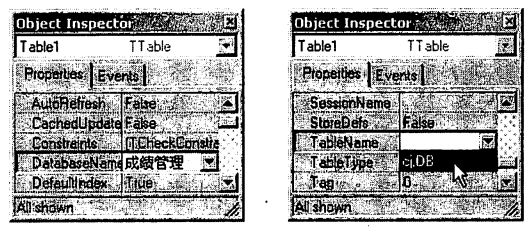


图 13-5 设置 TableName 属性

```
 table1.TableName:='cj.db';
 //...
end;
```

3. Active 属性

Active 属性用来设置是打开还是关闭与数据库之间的连接。设置为 True 表示连接数据库，设置为 False 表示关闭与数据库的连接。可以通过属性设置来实现；也可以通过程序代码来实现，如：Table1. Active :=True 或者 Table1. Active :=False；还可以使用 Table 的 Open/Close 方法，即：Table1. Open 与 Table1. Active :=True 的效果一样，Table1. Close 与 Table1. Active :=False 的效果一样。

Active 属性必须在 DatabaseName 属性和 TableName 属性设置之后再设置。如：

```
procedure TForm1.FormCreate(Sender: TObject);

 var curdir:string;

begin
 //...
 getdir(0,curdir);
 table1.Databasename:=curdir;
 table1.TableName:='cj.db';
 table1.Active:=True ;
 //...
end;
```

设置好上述三个属性之后，Table 就可以操作数据库表了。例如：

```
procedure TForm1.FormCreate(Sender: TObject);

 var curdir:string;

begin
 //...
 getdir(0,curdir);
 table1.Databasename:=curdir;
 table1.TableName:='cj.db';
 table1.Active:=true;
 table1.Edit;                                    //设置 Table 为编辑状态
 table1.FieldByName('姓名').Value:='张爱国';     //修改姓名为“张爱国”
 //...
end;
```

4. IndexName 属性

IndexName 属性用来索引名称，其可使表按照所设置的索引顺序显示记录。如果索引设置为空，则按照默认顺序排列。例如，Paradox 表按照主键顺序显示记录，而 VFP 表按照表的物理顺序显示等。

5. FieldCount 属性

FieldCount 属性用来返回表的字段的个数。

6. Fields 属性

Fields 属性是数组类型，Fields[n]表示表的第 n+1 个字段。如果不知道某个字段的名称，则可以使用这个属性，从而使用户通过循环的方法访问数据库表字段成为可能。

例如：

```
edit1.Text:=table1.Fields[0].DisplayName;        //显示字段名称“学号”
edit2.Text:=table1.Fields[0].Value;              //显示字段值“01”
edit3.Text:=table1.Fields[0].DisplayLabel ;      //显示字段名称“学号”
edit4.Text:=table1.Fields[0].FieldName;          //显示字段名称“学号”
```

7. RecordCount 和 RecNo 属性

RecordCount 属性用于返回与 Table 连接的表的记录个数，RecNo 属性用于设置或者返回当前的记录序号。

8. BOF 属性和 EOF 属性

BOF 属性用于返回当前是否是表的开始，值为 True 表示当前是表的开始。EOF 属性用于返回当前是否是表的结尾，值为 True 表示当前是表的结尾，值为 False 表示当前不是表的结尾。

注意：将记录指针(RecNo)指向最后一条记录 RecCount，此时 EOF 的值仍然为 False；如果将记录继续向后移动，此时 EOF 的值才会变成 True，而当前记录指针(RecNo)的值仍然保持为 RecCount。

【例 13-1】 浏览数据库表 cj.db。在窗体上添加 Memo 组件 Memo1、Table 组件 Table1、命令按钮组件 Button1、GroupBox 组件 GroupBox1。界面如图 13-6 所示。

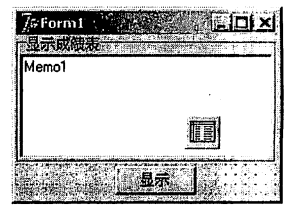


图 13-6 程序界面

分析：为了程序的可移植性，将应用程序和数据库表一起放在“D:\delphi 程序\ch13\eg13-1”中(请事先将数据库表 cj.db 复制到本文件夹)。

设计步骤如下：

(1) 添加组件，调整组件的大小和位置。

(2) 为了程序的可移植性，Table1 的属性不设置，在 Form 的 OnCreate 事件过程中编写如下代码。

```
var
 curdir:string;

procedure TForm1.FormCreate(Sender: TObject);
begin
 getdir(0,curdir);
```

```
 table1.DatabaseName:=curdir;
 table1.TableName:='cj.db';
 table1.Active:=true;
end;
```

(3) 编写 Button1 的 OnClick 事件过程，用于在 Memo1 中显示表的记录。代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);

 var i:integer;
    st:string;

begin
 memo1.Text:='学号    姓名    语文  数学  总分  名次';
 for i:=1 to table1.RecordCount do
 begin
  table1.RecNo:=i;                                    //指向第 i 条记录
  st:=table1.Fields[0].Text+'    ';                   //学号
  st:=st+table1.Fields[1].Text+'    ';                //姓名
  st:=st+inttostr(table1.Fields[2].Value) +'    ';    //语文
  st:=st+inttostr(table1.Fields[3].value)+'     ';    //数学
  st:=st+inttostr(table1.Fields[4].value)+'     ';    //总分
  st:=st+inttostr(table1.Fields[5].value);            //名次
  memo1.Lines.Add(st);
 end;
end;
```

(4) 运行程序，结果如图 13-7 所示。

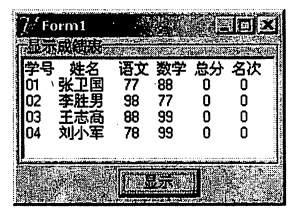


图 13-7　程序运行结果

请读者考虑下面两段代码哪个正确？

①

```
procedure TForm1.Button1Click(Sender: TObject);
 var st:string;

begin
 memo1.Text:='学号    姓名    语文  数学  总分  名次';
 table1.recno:=1;
 repeat
  st:=table1.Fields[0].Text+'    ';                   //学号
  st:=st+table1.Fields[1].Text+'    ';                //姓名
  st:=st+inttostr(table1.Fields[2].Value)+'    ';     //语文
  st:=st+inttostr(table1.Fields[3].value)+'     ';    //数学
  st:=st+inttostr(table1.Fields[4].value)+'     ';    //总分
  st:=st+inttostr(table1.Fields[5].value);            //名次
  memo1.Lines.Add(st);
  table1.recno:=table1.recno+1;
 until table1.RecordCount=table1.RecNo;
end;
```

②

```
procedure TForm1.Button1Click(Sender: TObject);

 var st:string;

begin
 memo1.Text:='学号    姓名    语文  数学  总分  名次';
 table1.recno:=1;
 repeat
  st:=table1.Fields[0].Text+'    ';                    //学号
  st:=st+table1.Fields[1].Text+'    ';                 //姓名
  st:=st+inttostr(table1.Fields[2].Value)+'    ';      //语文
  st:=st+inttostr(table1.Fields[3].value)+'      ';    //数学
  st:=st+inttostr(table1.Fields[4].value)+'      ';    //总分
  st:=st+inttostr(table1.Fields[5].value);             //名次
  memo1.Lines.Add(st);
  table1.recno:=table1.recno+1;
 until table1.Eof;
end;
```

9. Exclusive 属性

Exclusive 属性用于决定是否以独占方式打开数据库表。如果值为 True,则不允许其他用户访问数据库表;如果值为 False,则允许其他用户访问。

10. ReadOnly 属性

ReadOnly 属性用于是否以只读方式打开表,值为 True 表示以只读方式打开表,否则可改写表。修改此属性之前一定要先设置 Active 属性为 False,修改 ReadOnly 属性之后,再设置 Active 属性为 True。

11. Filter 属性和 Filtered 属性

Filter 属性用于设置过滤器。设置过滤器后,就只能操作那些满足条件的记录。Filtered 属性用于指定过滤器是否有效。例如:

```
table1.Filter:='学号=01';
table1.Filtered:=true;
```

可以用于显示学号为"01"的记录(假如其他属性已经设置好了)。

12. BookMark 属性

BookMark 属性是记录书签属性。书签是一种指针,它用于快速定位表记录。在程序中,通常需要将当前记录保存下来,在完成其他操作之后,再恢复表记录到原来的记录号。书签用法为:

```
var bt:tbookmarkstr;
 ...
 bt:=table1.Bookmark;          //操作前保存初始记录号到书签变量
 ...                           //其他操作
```

```
table1.Bookmark:=bt;                    //操作完毕,恢复表记录到初始记录号
```

13.1.2　Table 组件的常用方法

1. 移动记录指针的方法

First：将记录指针指向表的第 1 条记录。此时 BOF 的值为 True。

Prior：将记录指针指向前一条记录。如果当前是第 1 条记录，继续向前移动也不会出错，记录指针将仍然保持为 1。

Next：将记录指针指向表的下一条记录。如果当前是倒数第二条记录，执行 Next 之后，当前记录变成最后一条记录，此时 EOF 的值仍然为 False，如果继续执行 Next 则 EOF 变为 True，但是记录指针仍然为最后一条记录。

Last：将记录指针指向最后一条记录。同时置 EOF 的值为 True。

2. FieldByName 方法

该方法用于访问数据集中的某个字段，函数原型为：

```
function FieldByName(const FieldName: string): TField;
```

其中，参数 FieldName 是一个字符型参数，表示要引用的字段名。返回值是 Tfield 类型，通过返回类型可以得到指定字段的信息。使用该方法需要对字段的主要属性有所了解。

(1) Value 属性：用来返回或者设置当前记录对应字段的值。例如，table1. FieldByName('总分'). Value 表示当前记录的总分字段的值。

(2) FieldName 属性：返回对应字段的字段名。例如，例 13-2 中 table1. fieldbyname ('姓名'). FieldName 表示姓名字段，table1. Fields[1]. FieldName 也表示姓名字段。

(3) AsString 属性：把当前记录对应的字段的值转化为字符类型。类似的属性还有：AsVariant、AsBoolean、AsInteger、AsFloat、AsDateTime、AsCurrency、AsBCD 等。例如，例 13-2 中当前是第一条记录时 table1. fieldbyname('姓名'). AsString 的值就是“张卫国”。

(4) DataType 属性：返回字段的数据类型。数据类型如下：

type TfieldType = (ftUnknow, ftString, ftSmallint, ftInteger, ftWord, ftBoolean, ftFloat, ftCurrency, ftBCD, ftDate, ftTime, ftDateTime, ftBytes, ftVarBytes, ftAutoInc, ftBlob, ftMemo, ftGraphic, ftFmtMemo, ftParadoxOle, ftDBaseOle, ftTypedBinary, ftCursor, ftFixedChar, ftWideString, ftLargeint, ftADT, ftArray, ftReference, ftDataSet, ftOraBlobftOraClob, ftVariant, ftInterface, ftIDispatchftGuid, ftTimeStamp, ftFMTBcd)。

(5) FieldNo 属性：该属性返回字段在字段列表中的序号。序号从 1 开始。例如，table1. FieldByName('姓名'). FieldNo 的值是 2。

3. 与数据操作有关的方法

(1) Append 方法：在数据表末尾添加一条空记录，并将记录指针指向该记录。

(2) Insert 方法：在当前记录之后插入一条记录，并将记录指针指向该记录。

(3) Delete 方法：删除当前记录，并将记录指针指向下一条记录。

(4) Edit 方法：将数据表置为编辑状态，使用户可以修改数据表。

(5) Post 方法：修改的数据或者添加的数据更新。

(6) Cancel 方法：取消修改或者添加。

(7) Refresh 方法：刷新界面，界面上显示最新的数据。

(8) EmptyTable 方法：将数据表清空，保留数据表结构。

【例 13-2】 计算表 cj.db 的总分和名次。

分析：为了与例 13-1 分开，将上题中的数据库文件复制到“D:\delphi 程序\eg13-2”。为了使程序更加简洁，将计算总分和计算名次分开进行。

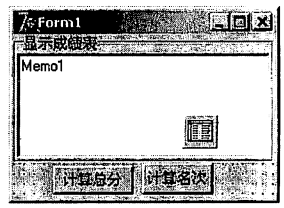


图 13-8 程序界面

设计步骤如下：

(1) 添加 GroupBox 组件 GroupBox1、Memo 组件 Memo1、Table 组件 Table1、命令按钮组件 Button1 和 Button2。界面如图 13-8 所示。

(2) 属性设置(省略)。

(3) 编写 Form 的 OnCreate 事件过程如下：

```
var
 curdir:string;

procedure TForm1.FormCreate(Sender: TObject);
begin
 getdir(0,curdir);
 table1.DatabaseName:=curdir;
 table1.TableName:='cj.db';
 table1.Active:=true;
end;
```

(4) 编写 Button1 的 OnClick 事件过程，用于计算总分，过程如下：

```
procedure TForm1.Button1Click(Sender: TObject);

 var st:string;

begin
 table1.First;                                          //从第 1 条记录开始
 while not table1.Eof do                                //到最后一条记录
 begin
  table1.Edit;                                          //编辑状态
  table1.FieldByName('总分').Value:=table1.FieldByName('语文').Value
  +table1.FieldByName('数学').Value;                    //计算总分
  table1.Post;                                          //计算总分有效
  table1.Next;                                          //下一条记录
 end;
 memo1.Text:='学号    姓名    语文  数学  总分  名次';
 table1.First;                                          //指向第 1 条记录
```

```
 repeat
  st:=table1.Fields[0].Text+'    ';                    //学号
  st:=st+table1.Fields[1].Text+'    ';                 //姓名
  st:=st+inttostr(table1.Fields[2].Value)+'    ';      //语文
  st:=st+inttostr(table1.Fields[3].value)+'     ';     //数学
  st:=st+inttostr(table1.Fields[4].value)+'     ';     //总分
  st:=st+inttostr(table1.Fields[5].value);             //名次
  memo1.Lines.Add(st);
  table1.Next;                                         //指向下一条记录
 until table1.Eof;
end;
```

(5) 编写 Button2 的 OnClick 事件过程,用于计算名次,代码如下:

```
procedure TForm1.Button2Click(Sender: TObject);

 var
  st:string;

begin
 table1.IndexName:='zf';                    //按照总分降序排列,便于计算总分
 table1.First;                              //指向第 1 条记录,即总分最高的记录
 while not table1.Eof do
 begin
  table1.Edit;
  table1.FieldByName('名次').Value:=table1.RecNo;      //名次与记录号相同
  table1.Post;
  table1.Next;
 end;
 table1.IndexName:='';                                 //重新按照主键顺序显示
 memo1.Text:='学号    姓名    语文  数学  总分  名次';
 table1.First;
 repeat
  st:=table1.Fields[0].Text+'    ';                    //学号
  st:=st+table1.Fields[1].Text+ '    ';                //姓名
  st:=st+inttostr(table1.Fields[2].Value)+ '    ';     //语文
  st:=st+inttostr(table1.Fields[3].value)+ '     ';    //数学
  st:=st+inttostr(table1.Fields[4].value)+ '     ';    //总分
  st:=st+inttostr(table1.Fields[5].value);             //名次
  memo1.Lines.Add(st);
  table1.Next;
 until table1.Eof;
end;
```

(6) 程序运行结果如图 13-9 所示。

4. 与查找有关的方法

(1) GotoKey 查找方法:此方法是一个函数。原型如下:

```
function GotoKey: Boolean;
```

该方法用于精确查找,被查找的字段必须是索引字段或是主键字段。查找步骤如下:

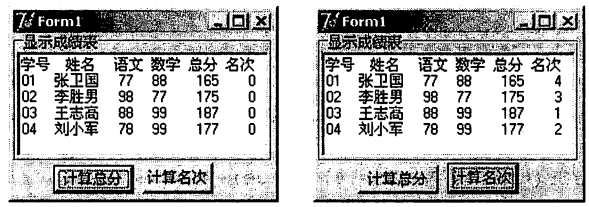


图 13-9 程序运行结果

① 设置查找索引，如果为空则表示按照主键查找。

② 执行 SetKey 方法置查找状态。

③ 设置查找关键字值。

④ 执行 GotoKey 进行查找。如果 GotoKey 方法返回值 True 表示查找成功，如果查找成功则将记录指针指向这条记录，如果查找失败则记录指针保持不变，同时返回 False。

(2) GotoNearest 查找方法：此方法用于模糊查找，没有返回值。处理过程类似于 GotoKey。原型为：

```
procedure GotoNearest;
```

(3) FindKey 查找方法：此方法属于索引查找，也是精确查找。函数原型为：

```
function FindKey(const KeyValues: array of const): Boolean; virtual;
```

FindKey 方法与 GotoKey 类似，区别在于 FindKey 查找是带参数的，参数就是要查找的值，参数可以有多列，多列参数用逗号隔开。查找成功返回 True，并指向该记录；否则返回 False，记录指针不变。

(4) FindNearest 查找方法：此方法是一个过程，用于模糊查找。由于终会查找成功，因此没有返回值，原型为：

```
procedure FindNearest(const KeyValues: array of const);
```

处理过程类似于 FindKey。

【例 13-3】 用多种方法查找数据库表 cj.db。

设置步骤如下：

(1) 将例 13-1 中的数据库表文件复制到“d:\delphi 程序\ch13\eg13-3”目录中。

(2) 添加组件，调节组件的大小和位置，设置属性(省略)。设计界面如图 13-10 所示。

(3) 编写代码。

① 编写 Form 的 OnCreate 事件过程代码如下：

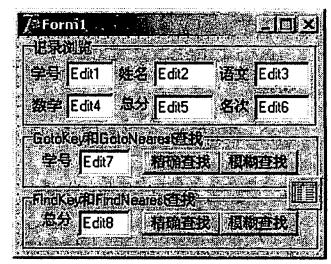


图 13-10 程序界面

```
var
 curdir:string;

procedure TForm1.FormCreate(Sender: TObject);
begin
 getdir(0,curdir);
 table1.DatabaseName:=curdir;
 table1.TableName:='cj.db';
```

```
 table1.Active:=true;
end;
```

② 编写 Button1 的 OnClick 事件过程，完成 GotoKey 查找。代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 table1.IndexName:='';       //按照主键(即：学号)查找,此语句可以省略
 table1.SetKey;
 table1.FieldByName('学号').AsString:=edit7.Text;
 if table1.GotoKey then
  begin
    edit1.Text:=table1.FieldByName('学号').Value;
    edit2.Text:=table1.FieldByName('姓名').Value;
    edit3.Text:=inttostr(table1.fieldbyname('语文').Value);
    edit4.Text:=inttostr(table1.fieldbyname('数学').Value);
    edit5.Text:=inttostr(table1.fieldbyname('总分').Value);
    edit6.Text:=inttostr(table1.fieldbyname('名次').Value)
  end
 else
  showmessage('查无此人');
end;
```

运行结果如图 13-11 所示。

③ 编写 Button2 的 OnClick 事件过程，完成 GotoNearest 查找。代码如下：

```
procedure TForm1.Button2Click(Sender: TObject);
begin
 table1.IndexName:='';       //按照主键学号查找,此语句可以省略
 table1.SetKey;
 table1.FieldByName('学号').AsString:=edit7.Text;
 table1.GotoNearest;
 edit1.Text:=table1.FieldByName('学号').Value;
 edit2.Text:=table1.FieldByName('姓名').Value;
 edit3.Text:=inttostr(table1.fieldbyname('语文').Value);
 edit4.Text:=inttostr(table1.fieldbyname('数学').Value);
 edit5.Text:=inttostr(table1.fieldbyname('总分').Value);
 edit6.Text:=inttostr(table1.fieldbyname('名次').Value)
end;
```

运行结果如图 13-12 所示。

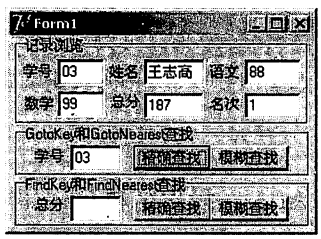


图 13-11　GotoKey 查找

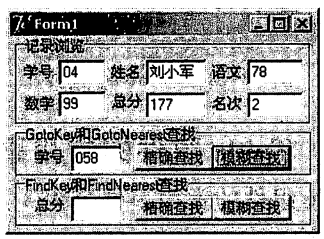


图 13-12　GotoNearest 查找

④ 编写 Button3 的 OnClick 事件过程，完成 FindKey 查找。代码如下：

```
procedure TForm1.Button3Click(Sender: TObject);
begin
 table1.IndexName:='zf';
 if table1.FindKey([edit8.Text]) then
  begin
   edit1.Text:=table1.FieldByName('学号').Value;
   edit2.Text:=table1.FieldByName('姓名').Value;
   edit3.Text:=inttostr(table1.fieldbyname('语文').Value);
   edit4.Text:=inttostr(table1.fieldbyname('数学').Value);
   edit5.Text:=inttostr(table1.fieldbyname('总分').Value);
   edit6.Text:=inttostr(table1.fieldbyname('名次').Value)
  end
 else showmessage('查无此人');
end;
```

运行程序，结果如图 13-13 所示。

⑤ 编写 Button4 的 OnClick 事件过程，完成 FindNearest 查找。代码如下：

```
procedure TForm1.Button4Click(Sender: TObject);
begin
 table1.IndexName:='zf';
 table1.FindNearest([edit8.Text]);
 edit1.Text:=table1.FieldByName('学号').Value;
 edit2.Text:=table1.FieldByName('姓名').Value;
 edit3.Text:=inttostr(table1.fieldbyname('语文').Value);
 edit4.Text:=inttostr(table1.fieldbyname('数学').Value);
 edit5.Text:=inttostr(table1.fieldbyname('总分').Value);
 edit6.Text:=inttostr(table1.fieldbyname('名次').Value)
end;
```

程序运行结果如图 13-14 所示。

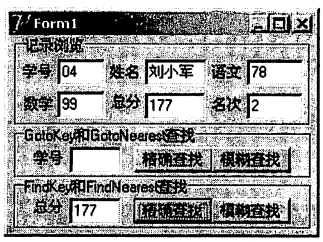


图 13-13 FindKey 查找

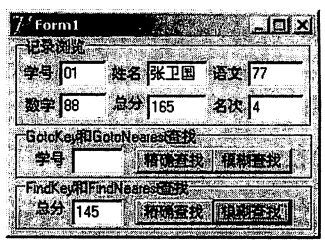


图 13-14 FindNearest 查找

13.1.3 Table 组件的常用事件

Table 组件的常用事件分为 3 类：Before＋操作名、After＋操作名和 On＋操作名，如表 13-1 所示。

表 13-1　Table 组件的常用事件

事　件	说　明
BeforeOpen、AfterOpen	发生在数据集打开前后
BeforeClose、AfterClose	发生在数据集关闭前后
BeforeInsert、AfterInsert	发生在添加记录前后
BeforeEdit、AfterEdit	发生在编辑记录前后
BeforePost、AfterPost	发生在写数据集前后
BeforeCancel、AfterCancel	发生在取消前后
BeforeDelete、AfterDelete	发生在删除记录前后
OnNewRecord	发生在创建新记录的时候
OnCalcField	发生在计算字段的时候

13.2　数据源 DataSource 组件

13.2.1　DataSource 组件的常用属性

1. AutoEdit 属性

AutoEdit 属性决定是否允许数据控制组件对数据进行修改。当取值为 True 时，允许修改；取值为 False 时，只有调用 Edit 才可以修改。

2. DataSet 属性

DataSet 属性指定为其提供数据集的组件，如 Table 组件、Query 组件和 StoredProc 组件。

3. State 属性

State 属性返回数据集组件的当前状态。

13.2.2　DataSource 组件的常用方法

1. Edit 方法

决定数据集中的数据是否可以编辑。

2. IsLinkedTo 方法

判断数据源是否与参数 DataSet 中指定的数据集相联系。

13.3　数据控制类组件

13.3.1　数据控制类组件的常用属性

1. DataSource 属性

该属性用来设置与本组件相连接的 TDataSource 组件。当在属性窗口中选定该属性

后，在列表框中将显示出所有的 TdataSource 组件供用户选择。也可以用代码实现该属性的设置。

2. DataField 属性

该属性决定着数据控制组件显示的字段名。设置 DataSource 属性，当在属性窗口中选中该属性后，在列表框中将显示出所有的字段名供用户选择。也可以用代码实现该属性的设置。

13.3.2 常用的数据控制类组件

常用的数据控制类组件主要用来显示与修改数据记录，包含 DBEdit 组件、DBGrid 组件、DBMemo 组件、DBNavigator 组件、DBText 组件、DBListBox 组件、DBComboBox 组件、DBCheckBox 组件、DBRadioGroup 组件等。

1. DBNavigator 组件

DBNavigator 组件是数据导航组件，如图 13-15 所示。

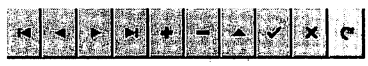


图 13-15 DBNavigator 组件

DBNavigator 组件各个按钮的功能如表 13-2 所示。

表 13-2 DBNavigator 组件按钮及功能一览表

图标	按钮名称	按钮功能
	First	将数据集记录指针指向第 1 条记录，相当于 First 方法
	Prior	将数据集记录指针指向上一条记录，相当于 Prior 方法
	Next	将数据集记录指针指向下一条记录，相当于 Next 方法
	Last	将数据集记录指针指向最后一条记录，相当于 Last 方法
	Insert	插入一条记录，相当于 Insert 方法
	Delete	删除当前记录，相当于 Delete 方法
	Edit	修改记录，相当于 Edit 方法
	Post	保存修改，相当于 Post 方法
	Cancel	取消修改，相当于 Cancel 方法
	Refresh	刷新显示，相当于 Refresh 方法

DBNavigator 组件的主要属性如下。

(1) VisibleButtons 属性：选择设置 DBNavigator 组件的各个按钮是否显示。该属性包含多个可选项。当不需要那么多按钮的时候，可以将某些按钮设置为不显示。

(2) ShowHint 属性：控制是否显示各个按钮的动态提示信息。默认值为 False 时，表示不显示动态提示信息；值为 True 表示显示按钮的动态提示信息。

(3) Hints 属性：设置各个按钮的动态显示信息。用户设置的按钮动态信息会覆盖默认的按钮动态信息。

(4) ConfirDelete 属性：删除记录前是否给出确认信息。如果值为 True，则在删除记录前提示是否删除；值为 False 时直接删除记录，不做任何提示。

2. 其他组件

这里主要指 DBEdit 组件、DBGrid 组件、DBMemo 组件、DBText 组件、DBListBox 组件、DBComboBox 组件、DBCheckBox 组件、DBRadioGroup 组件、DBImage 组件等。只需要设置 DataSource 和 DataField 属性即可使用上述组件。

新建文件夹“D:\delphi 程序\ch13\eg13-4”，然后在此文件夹中新建一个教师档案表 jsda.db，并输入若干条记录，如图 13-16 所示。新建过程省略。

【例 13-4】 教师档案管理系统。要求使用数据控制类组件和数据源组件。

设计步骤如下：

(1) 在“D:\delphi 程序\ch13\eg13-4”中新建应用程序。界面如图 13-17 所示。

	Field Name	Type	Size	Key
1	工号	A	2	*
2	姓名	A	6	
3	性别	A	2	
4	职称	A	6	
5	是否党员	L		
6	照片	G		
7	备注	M	100	

图 13-16　jsda.db 的表结构

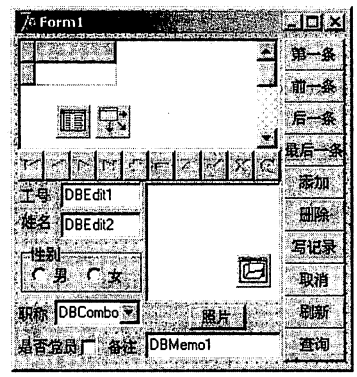


图 13-17　程序界面

(2) 属性设置见表 13-3 所示。

表 13-3　教师档案管理系统组件属性设置

对　象	属　性	属　性　值	说　明
Table1	DatabaseName	D:\delphi 程序\ch13\eg13-4	Table 组件连接到表文件 jsda.db
	TableName	jsda.db	
	Active	True	
DataSource1	DataSet	Table1	连接数据集组件 Table1
DBGrid1	DataSource	DataSource1	显示数据源 DataSource1 连接的数据集
DBNavigator	DataSource	DataSource1	为 DataSource1 连接的数据源导航
DBEdit1	DataSource	DataSource1	显示工号字段
	DataField	工号	

续表

对　象	属　性	属　性　值	说　明
DBEdit2	DataSource	DataSource1	显示姓名字段
	DataField	姓名	
DBGroup1	DataSource	DataSource1	显示性别字段
	DataField	性别	
	Caption	性别	
	Lines[0]	男	
	Lines[1]	女	
DBComboBox1	DataSource	DataSource1	显示职称字段
	DataField	职称	
DBCheckBox1	DataSource	DataSource1	显示是否党员字段
	DataField	是否党员	
	Caption	是否党员	
DBImage1	DataSource	DataSource1	显示照片字段
	DataField	照片	
DBMemo1	DataSource	DataSource1	显示备注信息
	DataField	备注	
Button1	Caption	照片	用于添加或者修改照片字段
Button2	Caption	第一条	
Button3	Caption	前一条	
Button4	Caption	后一条	
Button5	Caption	最后一条	
Button6	Caption	添加	
Button7	Caption	删除	
Button8	Caption	写记录	
Button9	Caption	取消	
Button10	Caption	刷新	
Button11	Caption	查询	
Label1～Label4	Caption	工号、姓名、职称、备注	

(3) 编写代码如下：

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 DBCombobox1.Items.Add('助教');
```

```
 DBComboBox1.Items.Add('讲师');
 DBCombobox1.Items.Add('副教授');
 DBCombobox1.Items.Add('教授');
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
 table1.Edit;                    //用于编辑,此处为添加或修改照片
 if openpicturedialog1.Execute then
 dbimage1.Picture.LoadFromFile(openpicturedialog1.FileName);
 table1.Post;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
 table1.First;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
 table1.Prior;
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
 table1.Next;
end;

procedure TForm1.Button5Click(Sender: TObject);
begin
 table1.Last;
end;

procedure TForm1.Button6Click(Sender: TObject);
begin
 table1.Append;
end;

procedure TForm1.Button7Click(Sender: TObject);
begin
 table1.Delete;
end;

procedure TForm1.Button8Click(Sender: TObject);
begin
 table1.Post;
end;

procedure TForm1.Button9Click(Sender: TObject);
begin
 table1.Cancel;
```

```
end;

procedure TForm1.Button10Click(Sender: TObject);
begin
 table1.Refresh;
end;

procedure TForm1.Button11Click(Sender: TObject);          //查询
 var gh:string;
begin
 gh:=inputbox('请输入要查询的工号!','输入工号','');
 table1.IndexName:='';
 table1.SetKey;
 table1.FieldByName('工号').AsString:=gh;
 if not table1.GotoKey then
  showmessage('查无此人!')
end;
```

说明：其实不用编写任何代码，本程序已经可以正常运行，并且可以显示数据库表记录，还可以对数据库表进行除了照片操作之外的任何操作。

为了程序的可移植性，用户可以先设置好包括 Table1 组件、DataSource1 组件在内的所有组件的属性，然后再取消对 Table1 组件的 3 个属性(DatabaseName、TableName 和 Active)的设置，然后在 Form 的 OnCreate 事件过程中再来设置这 3 个属性。代码如下：

```
procedure TForm1.FormCreate(Sender: TObject);

 var curdir:string;

begin
 getdir(0,curdir);
 Table1.DatabaseName:=curdir;
 table1.TableName:='jsda.db';
 table1.Active:=true;
 DBCombobox1.Items.Add('助教');
 DBComboBox1.Items.Add('讲师');
 DBCombobox1.Items.Add('副教授');
 DBCombobox1.Items.Add('教授');
end;
```

说明：先设置 Table1 组件、DataSource1 组件的属性是为了方便设置控制类组件的其他各个属性。所有属性设置完毕再取消 Table1 组件的 3 个属性设置，是为了在 Form 的 OnCreate 事件过程中动态设置 Table1 的数据库别名，这样数据库别名将不受路径限制可以移植到其他电脑。

(4) 运行程序，结果如图 13-18 所示。

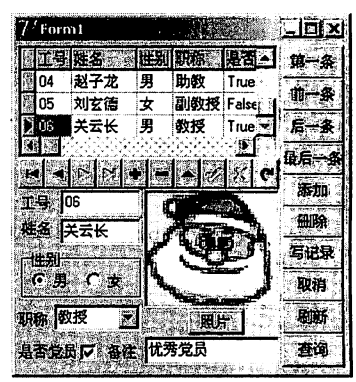


图 13-18 程序运行结果

13.4　Query 组件

和 Table 组件一样，Query 组件是 BDE 中又一个非常重要的组件。与 Table 组件不同的是，Table 组件与一个具体的数据表相连，而 Query 组件则与整个数据库相连，因此 Query 组件可以访问一个表或者多个表。

Query 组件可以将 SQL 语句传递到数据库系统的数据库引擎中，由数据库引擎具体执行 SQL 语句，以实现对数据库的具体操作。使用 Query 组件能够实现如下具体功能。

(1) 添加、修改、删除数据。

(2) 静态查询。

(3) 实现带有变量的动态查询。

(4) 实现带有参数的动态查询。

13.4.1　Query 组件的常用属性

1. DatabaseName 属性

该属性用来指明 Query 组件要访问的数据库别名(或数据库路径)。设置方法与 Table 组件类似。

2. SQL 属性

Table 组件通过 TableName 访问某个数据库表，而 Query 组件则通过 SQL 属性访问某个数据库。SQL 字符串中存放着 SQL 语句，设置好 DatabaseName 属性后，再设置 SQL 属性，然后设置 Active 属性为 True，运行程序，即可执行 SQL 属性中指定的 SQL 语句。

3. Active 属性

该属性设置为 True 时，可执行 SQL 属性中指定的 SQL 语句。

13.4.2　Query 组件的常用方法

1. Open 方法

该方法打开或者激活 Query 组件的数据集。执行该方法后，将执行 SQL 属性中指定的 SQL 语句(只能是 Select 语句)，结果返回一个数据集。

2. ExecSQL 方法

该方法和 Open 方法类似，也执行 SQL 中指定的 SQL 语句，但 Open 只能执行 Select 语句操作，而 ExecSQL 则能进行添加(Insert)、删除(Delete)、更新(Update)等操作。因此，如果要执行 Select 语句，习惯上使用 Open 方法，但是要执行添加(Insert)、删除(Delete)、更新(Update)等操作则必须使用 ExecSQL 方法。

3. Close 方法

该方法用于关闭数据集。

4. Prepare 方法

该方法将 SQL 语句送到数据库引擎，由数据库引擎对 SQL 语句进行分析，从而可以大大提高动态 SQL 语句的执行效能。

5. SQL 方法

SQL 有两个方法可以添加或清空 SQL 语句。

(1) Add 方法：该方法用于向 SQL 中添加 SQL 语句。

(2) Clear 方法：该方法用于清空 SQL 语句。一般来说，在执行 Add 方法之前要先清空 SQL 中先前的 SQL 语句。

例如：

```
Query1.Close;
Query1.SQL.Clear;
Query1.SQL.Add('select * from jsda.db';);
Query1.Open;
```

13.4.3 静态查询

有的查询其 SQL 语句在程序中是不变的，查询结果也是一定的，像这种查询就称为静态查询。实现静态查询有两种方法。第一种方法是直接设置所有属性，程序运行直接显示结果，无须编写任何代码。第二种方法是通过代码来实现查询。

【例 13-5】 编写程序显示 jsda. db 这个表文件。

设计步骤如下：

(1) 在窗体上添加 Query 组件 Query1、DataSource 组件 DataSource1、DBGrid 组件 DBGrid1，界面如图 13-19 所示。

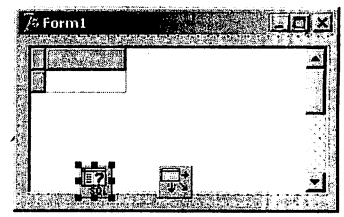


图 13-19 程序界面

(2) 先建立“D:\delphi 程序\ch13\eg13-5”，将 jsda. db 这个表复制到该文件夹中。设置属性，见表 13-4。

表 13-4 属性设置表

对 象	属 性	属 性 值
Query1	DatabaseName	D:\delphi 程序\ch13\eg13-5
	SQL	'Select * from jsda. db'
	Active	True
DataSource1	DataSet	Query1
DBGrid1	DataSource	DataSource1

其中，SQL 属性的设置方法是单击 Query1 的 SQL 属性后的按钮，弹出 String List Editor 对话框，在对话框中输入 SQL 语句，按 OK 按钮。如图 13-20 所示。

(3) 属性设置完毕，教师档案表的信息就已经显示出来，如图 13-21 所示。运行程序如图 13-22 所示。

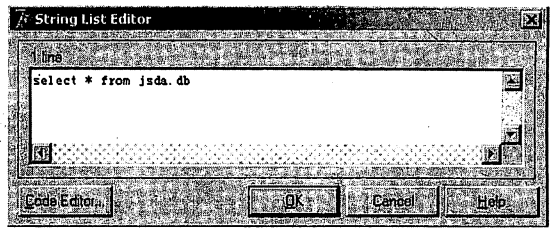


图 13-20　设置 SQL 属性

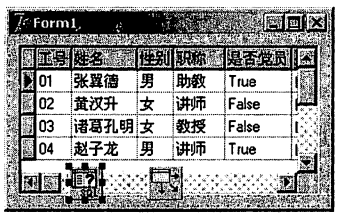


图 13-21　设置完属性后的界面

图 13-22　程序运行界面

为了后面例题的需要，将前面建立的数据库成绩表 cj. db 也复制到该文件夹下。再在“D:\delphi 程序\ch13\eg13-6”中新建表 cj1. db。cj. db 和 cj1. db 如图 13-23 所示。

【例 13-6】 编写程序，显示文件夹“D:\delphi 程序\ch13\eg13-6”中 cj. db 和 cj1. db 中每个人的学号、姓名和各科成绩。

(1) 在窗体上添加 Query 组件 Query1、DataSource 组件 DataSource1、DBGrid 组件 DBGrid1，界面如图 13-24 所示。

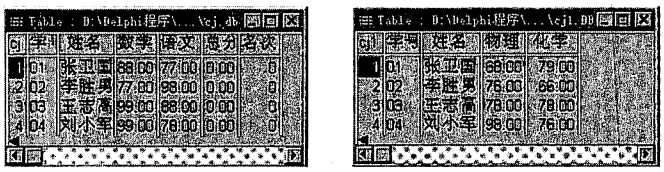


图 13-23　“D:\delphi 程序\ch13\eg13-6”下的两个表文件

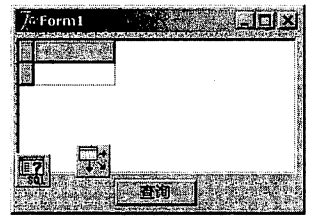


图 13-24　程序界面

(2) 设置各个组件的属性，如表 13-5 所示。

表 13-5　属性设置

对　象	属　性	属 性 值
Query1	DatabaseName	D:\delphi 程序\ch13\eg13-6
	SQL	
	Active	False
DataSource1	DataSet	Query1
DBGrid1	DataSource	DataSource1
Button1	Caption	查询

(3) 编写代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 query1.Close;
 query1.SQL.Clear;
 query1.SQL.Add('select 学号,姓名,语文,数学,物理,');
 query1.SQL.Add('化学 from cj,cj1 where cj1.学号=
   cj.学号');
 query1.Open;
end;
```

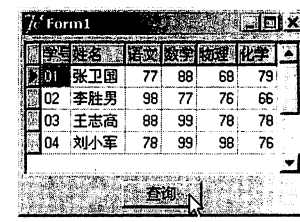


图 13-25 程序运行结果

(4) 运行程序，如图 13-25 所示。

13.4.4 使用字符连接符“+”实现动态查询

“+”是字符连接运算符，用于连接两个或多个字符串。在使用查询的时候可以使用“+”来把变量的值连接到 Query 组件的 SQL 属性中。例如，在 cj.db 表中，我们要查询的学号来自于编辑框 Edit1，要查询该学号对应学生的成绩。代码可以写成：

```
query1.SQL.Add('select * from cj where 学号='+''''+edit1.Text+'''');
```

上面的字符表达式可以理解为两个字符串：'select * from cj where 学号='和''''+edit1.Text+''''，这两个字符串通过“+”相连接。其中''''+edit1.Text+''''最外面的那对单引号里面的内容是'''+edit1.Text+'''，这 3 对单引号和两个“+”用于置换出 edit1 的内容。

【例 13-7】 设计一个数据库应用程序。该程序在编辑框中输入学号，按“查询”按钮可以查询 cj.db 中该学生的成绩记录。

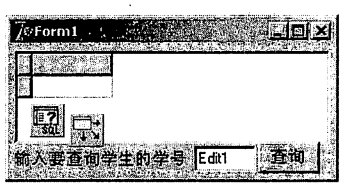


图 13-26 程序界面

设计步骤如下：

(1) 复制 cj.db 表到文件夹“D:\delphi 程序\ch13\eg13-7”中。新建应用程序，在窗体上添加组件。界面如图 13-26 所示。

(2) 设置属性，见表 13-6。

表 13-6 属性设置

对 象	属 性	属 性 值
Query1	DatabaseName	D:\delphi 程序\ch13\eg13-7
	SQL	
	Active	False
DataSource1	DataSet	Query1
DBGrid1	DataSource	DataSource1
Button1	Caption	查询
Label1	Caption	输入要查询学生的学号

(3) 编写代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 query1.Close;
 query1.SQL.Clear;
 query1.SQL.Add('select * from cj where ');
 query1.SQL.Add('学号='''+edit1.Text+'''');
 query1.Open;
end;
```

(4) 运行程序，结果如图 13-27 所示。

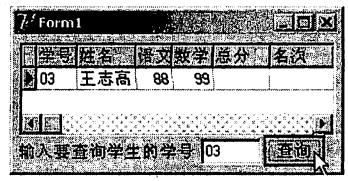


图 13-27　程序运行结果

13.4.5　使用 Params 属性实现参数查询

Query 组件有一个属性 Params，该属性在运行的时候访问，在设计程序时不可以使用。该属性是一个程序运行时动态建立的数组，每个数组元素分别对应一个 SQL 参数。例如：

```
Select * from cj where 学号>=:xh1 and 学号<=:xh2
```

这里有两个参数 xh1 和 xh2，分别对应两个数组元素 Params[0]和 Params[1]。使用语句：

```
Query1.Params[0].Value:=Edit1.Text;
Query1.Params[1].Value:=Edit2.Text;
```

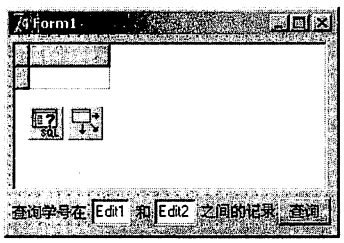


图 13-28　程序界面

以及上面的 SQL 语句可以实现动态查询。

【例 13-8】 设计一个数据库应用程序。该程序在编辑框中输入学号，按“查询”按钮可以查询 cj.db 中该学生的成绩记录。

设计步骤如下：

(1) 复制 cj.db 表到文件夹 D:\delphi 程序\ch13\eg13-8 中。新建应用程序，在窗体上添加组件。界面如图 13-28 所示。

(2) 设置属性，见表 13-7。

表 13-7　属性设置

对　象	属　性	属　性　值
Query1	DatabaseName	D:\delphi 程序\ch13\eg13-8
	SQL	
	Active	False
DataSource1	DataSet	Query1
DBGrid1	DataSource	DataSource1

续表

对　象	属　性	属 性 值
Button1	Caption	查询
Label1	Caption	查询学号在
Label2	Caption	和
Label3	Caption	之间的记录

(3) 编写代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 query1.Close;
 query1.SQL.Clear;
 query1.SQL.Add('select * from cj where 学号>=:xh1 and 学号<=:xh2');
 query1.Params[0].Value:=edit1.Text;
 query1.Params[1].Value:=edit2.Text;
 query1.Open;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
 query1.Close;
 query1.SQL.Clear;
 query1.SQL.Add('select * from cj');
 query1.Open;
end;
```

(4) 运行程序，结果如图 13-29 所示。

上述查询也可以这样进行：先设置好 Query1 的 SQL 属性为 select * from cj where 学号>=: xh1 and 学号<=: xh2，然后在程序中执行语句。

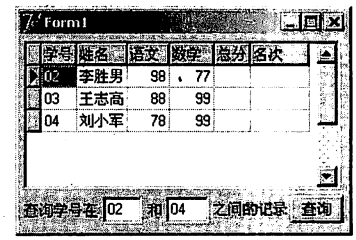
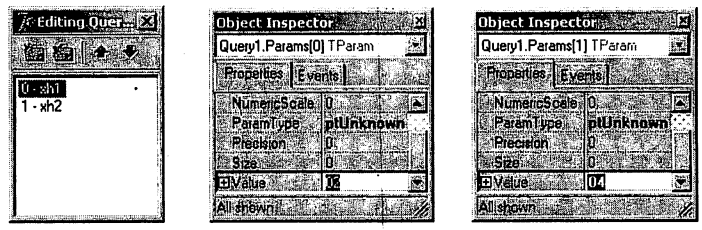


图 13-29　程序运行结果

```
query1.Close;
query1.Params[0].Value:=edit1.Text;
query1.Params[1].Value:=edit2.Text;
query1.Open;
```

还可以使用静态的方法。首先设置好 Query1 的 SQL 属性为 select * from cj where 学号>=: xh1 and 学号<=: xh2，然后设置 Query1 的 Params 属性，如图 13-30(a)所示。

分别选择 xh1 和 xh2，设置 xh1 的 Value 值和 xh2 的 Value 值，如图 13-30(b)、(c)所示。再设置 Query1 的 Active 属性值为 True，即可显示满足条件的记录。



(a)　(b)　(c)

图 13-30　SQL 属性与 Params 属性设置

13.4.6　使用 ParamByName 属性实现参数查询

Query 组件有一个属性 ParamByName，该属性与 Params 属性类似。格式为：

```
function ParamByName(const Value: String): TParam;
```

其中，Value 是字段名称，需要用引号引起来，返回值是 Tparam 类型，需要用 Value 或 ASString 等来改变数据类型。

如果使用 ParamByName 属性，则例 13-8 的程序可以写成：

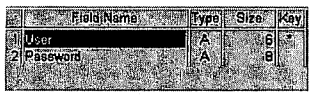
```
procedure TForm1.Button1Click(Sender: TObject);
begin
 query1.Close;
 query1.SQL.Clear;
 query1.SQL.Add('select * from cj where 学号>=:xh1 and 学号<=:xh2');
 query1.ParamByName('xh1').Value:=edit1.Text;
 query1.ParamByName('xh2').Value:=edit2.Text;
 query1.Open;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
 query1.Close;
 query1.SQL.Clear;
 query1.SQL.Add('select * from cj');
 query1.Open;
end;
```

13.5　使用 BDE 开发数据库综合实例

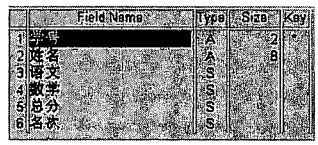
使用 BDE 组件建立一个成绩管理系统，要求该系统可以实现学生信息的添加、修改、录入、浏览、查询，并通过密码登录。

1. 数据库的建立

建立 Paradox 表文件“用户.db”和“成绩.db”，并把这两个表文件保存在事先建立好的文件夹“D:\Delphi 程序\BDE 综合”中。两个表的结构如图 13-31 所示。



(a) “用户.db” 表结构



(b) “成绩.db” 表结构

图 13-31 “用户.db”和“成绩.db”的表结构

其中,“用户.db”表文件的 User 字段设置有主键,“成绩.db”表文件的“学号”字段设置有主键,且为“总分”字段定义了第二索引,该索引是降序索引,名称为 zf。

使用 BDE 建立数据库别名为“BDE 综合”,Path 属性设置为“D:\Delphi 程序\BDE 综合”,如图 13-32 所示。

2. 界面设计

建立成绩管理系统界面。

首先新建应用程序 application。然后在此基础上再新建数据模块:选择菜单 File|New|Others...,在出现的 New Items 对话框中选择 New 选项卡,在 New 选项卡中选择 Data Module 选项,并按 OK 按钮,将数据模块改名为 dm。最后,再新建窗体 Form3:选择 File|New|Form。此时界面共有两个窗体(Form1 和 Form3),还有一个数据模块(dm)。

添加组件,在 dm 模块中添加组件 Tabel1 和 DataSource1、Table2 和 DataSource2,如图 13-33 所示。

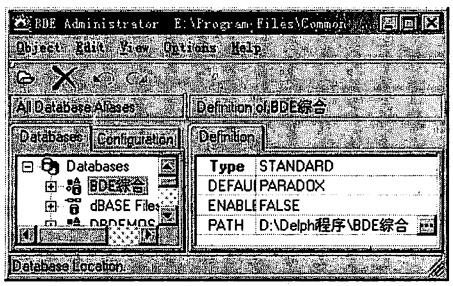


图 13-32 建立数据库别名

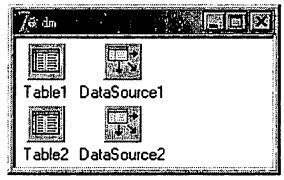


图 13-33 在 dm 中添加的组件

在 Form1 中添加组件 Label1、Label2、Label3、Edit1、DBLookUpComboBox1、BitBtn1 和 BitBtn2,界面如图 13-34 所示。

在 Form3 中添加组件 DBGrid1、DBNavigator1、GroupBox1、RadioButton1、RadioButton2、Edit1、Button1、Button2 和 Button3,界面如图 13-35 所示。

3. 属性设置

dm 模块中组件、窗体 Form1 中组件和窗体 Form3 中组件的属性设置分别如表 13-8、表 13-9 和表 13-10 所示。

图 13-34　在 Form1 中添加的组件

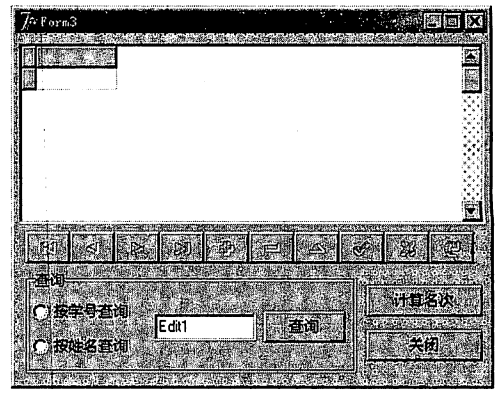


图 13-35　在 Form3 中添加的组件

表 13-8　dm 模块中组件的属性设置

对　象	属　性	属 性 值	说　明
Table1	DatabaseName	BDE 综合	为 Form1 中的组件提供数据源
	TableName	用户.db	
	Active	True	
DataSource1	DataSet	Table1	
Table2	DatabaseName	BDE 综合	为 Form3 中的组件提供数据源
	TableName	成绩.db	
	Active	True	
DataSource2	DataSet	Table2	

表 13-9　Form1 中组件的属性设置

对　象	属　性	属 性 值	说　明
Lable1	Caption	&user	设置热键
	FocusControl	DBLookupComboBox1	
Label2	Caption	&Password	设置热键
	FocusControl	Edit1	
Label3	Caption	成绩管理系统	
BitBtn1	Caption	取消	
BitBtn2	Caption	确定	
DBLookupComboBox1	ListSource	Dm. DataSource1	数据源来自 dm 模块中的 DataSource1
	KeyField	User	

表 13-10 Form3 中组件的属性设置

对 象	属 性	属 性 值	说 明
DBGrid1	DataSource	Dm. DataSource2	数据源来自 dm 模块中的 DataSource2
DBNavigator1	DataSource	dm. DataSource2	
GroupBox1	Caption	查询	
RadioButton1	Caption	按学号查询	
RadioButton2	Caption	按姓名查询	
Button1	Caption	查询	
Button2	Caption	计算名次	
Edit1	Text	设置为空	
Button3	Caption	关闭	

4. 编写代码

为登录窗体 Form1 中的“确定”按钮编写如下代码：

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
 if (dm.table1.fieldbyname('password').asstring=edit1.Text)
 then begin
      form3.show;
      form1.Release;          //释放 Form1
      end;
end;
```

为登录窗体 Form1 中的“取消”按钮编写如下代码：

```
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
 edit1.Text:='';
 edit1.SetFocus;
end;
```

为窗体 Form3 中的“查询”按钮 Button1 编写如下代码：

```
procedure TForm3.Button1Click(Sender: TObject);
begin
 if radiobutton1.Checked
 then begin
      dm.table2.IndexName:='';                          //按主键查询
      dm.table2.SetKey;
      dm.table2.FieldByName('学号').AsString:=edit1.Text;
      if not dm.table2.GotoKey
      then showmessage('查无此人');
     end
```

```
 else begin
      if not dm.table2.Locate('姓名',edit1.Text,[])        //按姓名查询
      then showmessage('查无此人');
    end;
end;
```

为窗体 Form3 中的“计算名次”按钮 Button2 编写如下代码：

```
procedure TForm3.Button2Click(Sender: TObject);
begin
 dm.table2.IndexName:='';                    //按主键顺序显示
 dm.Table2.First;                            //从第一条记录开始
 repeat                                      //循环计算每条记录的总分
  dm.Table2.Edit;
  dm.Table2.FieldByName('总分').Value:=dm.Table2.FieldByName('语文').Value
   +dm.Table2.FieldByName('数学').Value;
  dm.Table2.Post;
  dm.Table2.Next;
 until dm.Table2.Eof;
 dm.Table2.IndexName:='zf';                  //按总分降序显示以便计算名次
 dm.Table2.First;                            //指向总分为最高分的记录(因为是降序显
                                               示,所以第一条记录为最高分)
 repeat                                      //从最高分开始计算名次
  dm.Table2.Edit;
  dm.Table2.FieldByName('名次').Value:=dm.Table2.RecNo;
  dm.Table2.Post;
  dm.Table2.Next;
 until dm.Table2.Eof;
 dm.Table2.IndexName:='';                    //仍然按照主键顺序显示
 dm.Table2.First;
end;
```

为 Form3 中的“关闭”按钮 Button3 编写如下代码：

```
procedure TForm3.Button3Click(Sender: TObject);
begin
 application.Terminate ;                    //关闭应用程序
end;
```

为 Form3 的关闭事件编写如下代码：

```
procedure TForm3.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  application.Terminate ;                   //关闭应用程序
end;
```

运行程序，程序正常。至此，成绩管理系统编写完毕，保存程序的所有文件到文件夹“D:\Delphi 程序\BDE 综合”，这样便于维护程序。

说明：读者应该注意，本程序仅可以在指定的文件夹“D:\Delphi 程序\BDE 综合”下正常运行。读者若想在自己的计算机上正常运行，还需要使用 BDE 设置数据库别名。为

了让程序具有较强的可移植性，应该动态设置数据库别名（动态设置路径），这点请务必注意。

13.6 小结

本章是本书的重点，讲述了 BDE 数据库，具体内容如下。

- Table 组件，本节讲述了 Table 组件的属性、方法、事件以及 Table 组件的用法。
- DataSource 组件。
- 数据控制类组件，主要讲述了 DBEdit、DBMemo、DBGrid、DBNavigator 等组件。
- Query 组件，主要讲述了如何使用 Query 组件进行多表查询。
- 开发 BDE 数据库综合例题，本节讲述了使用 BDE 组件编写成绩管理系统。

本章中最重要的组件是 Table 组件和 Query 组件，读者应该学会使用 Table 组件操作单表，使用 Query 组件操作多表，还要学会灵活使用其他数据控制类组件。

习题

1. Table 的_____属性用来与数据库连接。

   A. DataBaseName　B. TableName　C. DataSource　D. SQL

2. DataSource 组件的_____属性用于指明数据集。

   A. DataBaseName　B. TableName　C. DataSet　D. SQL

3. 要使 Query 组件的 SQL 语句执行后返回一个结果数据集，应调用 Query 组件的_____方法。

   A. Add　B. Open　C. ExecSQL　D. Open 或者 ExecSQL

4. Open 方法和 ExecSQL 方法有何区别？

5. 什么是静态方法？什么是动态方法？

6. 编写程序，要求仅用 Table 组件和 Memo 组件显示 Paradox 表文件。

7. 利用数据库向导制作数据库应用程序，该应用程序用于操作“...\Common Files\Borland Shared\Data”中的 country.db 文件。

8. 设计一个数据库应用程序，要求使用 Query 组件，并使用 Params 属性实现对“...\Common Files\Borland Shared\Data”中 country.db 文件的查询。

9. 设计一个数据库应用程序，要求使用 Query 组件，并使用 ParamByName 属性实现对“...\Common Files\Borland Shared\Data”中 country.db 文件的查询。

第 14 章

ADO 数据库应用程序开发

ADO 是 Active Data Object 的简称，是微软公司面向各种数据的高层接口。ADO 可以快速访问各种数据库资源，它是一个高性能、高兼容性的数据访问接口。

在 Delphi 中有一个组件面板 ADO，它提供了一些组件，使用这些组件可以访问各种数据库资源。ADO 面板如图 14-1 所示。这些组件包括 ADOConnection 组件、ADOCommand 组件、ADODataSet 组件、ADOTable 组件和 ADOQuery 组件等。

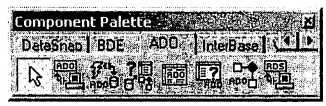


图 14-1　ADO 组件

ADO 组件与 BDE 组件具有类似的功能。ADO 组件与 BDE 组件的对应关系如表 14-1 所示。

表 14-1　ADO 组件与 BDE 组件的对应关系

类　别	连接组件	表 组 件	查询组件
ADO 组件	ADOConnection	ADOTable	ADOQuery
BDE 组件	DataSource	Table	Query

14.1　ADOConnection 组件

ADOConnection 组件的作用是建立与数据库的连接，此外它还有一定的事务处理能力。该连接可以被多个数据集共享，但并不是应用程序必须的，因为 ADO 数据集组件和 ADO 命令组件可以通过设置 ConnectionString 直接连接到数据库。不过最好还是使用 ADOConnection 组件，因为它起到了共同的桥梁作用，这样可以避免程序中的其他组件都要设置 ConnectionString 属性的烦琐。ADOConnection 组件具有如下功能。

(1) 连接数据库。

(2) 控制服务器注册。

(3) 某些事务管理。

(4) 为关联的组件提供数据库连接。

(5) 将 SQL 命令发送到数据库。

14.1.1 ADOConnection 组件的常用属性

1. CommandCount 属性

该属性用于确定有多少个 ADOCommand 组件与该 ADOConnection 组件关联。例如,下面的程序将执行与该组件连接的 ADOCommand 组件中的命令。

```
var i:integer;
begin
 for i:=0 to ADOConnection1.CommandCount-1 do
  ADOConnection1.Commands[i].Execute;
end;
```

2. Commands 属性

该属性是一个数组,它列出了 ADOConnection 组件连接的所有活动的 ADOCommand 组件名称。它的每一个成员代表一个活动的 ADOCommand 组件。

3. Connected 属性

该属性是 Boolean 类属性,用来确定连接是否是活动的,如果是活动的则返回 True,否则返回 False。

4. ConnectionString 属性

该属性是一个可读写的 String 类型属性,用于连接数据。有如下两种连接方式给出连接字符串。

(1) 直接给 ConnectionString 属性赋值。ConnectionString 字符串中有 Provider、DataSource、Persist Security Info、Extended Properties 等内容。例如:

```
Provider=MSDASQL.1;
User ID=hdh;Extended Properties="DSN=MS Access Database;
DBQ=D:\数据库 9\cj.db;
DefaultDir=D:\data9; UID=admin;"
...
```

(2) 上面的字符串读者读起来会感觉很烦琐,其实还可以使用属性编辑器给 ConnectionString 属性赋值。假设作者计算机上的"d:\数据库 9"下有文件 cj.db,双击 ConnectionString 属性后的按钮,出现属性编辑框,如图 14-2 所示。

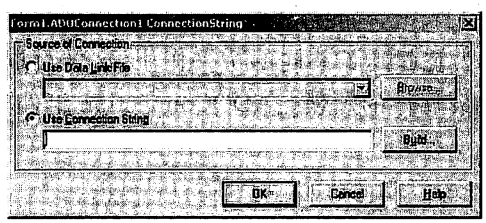


图 14-2 ConnectionString 属性编辑框

单击 Build 按钮，出现如图 14-3 所示对话框。

在对话框中选择 Microsoft Jet 4.0 OLE DB Provider，然后选择“所有”选项卡，如图 14-4 所示。选择“编辑值...”按钮，出现如图 14-5 所示对话框。在对话框的“属性值”中输入 Paradox 7.x，并单击“确定”按钮，出现如图 14-6所示对话框。设置数据库名为 cj.db 所在的文件夹“d:\数据库 9”，并将“用户名称”右边的编辑框内容去掉。单击“测试连接”按钮，出现“测试连接成功”字样，如图 14-7所示。

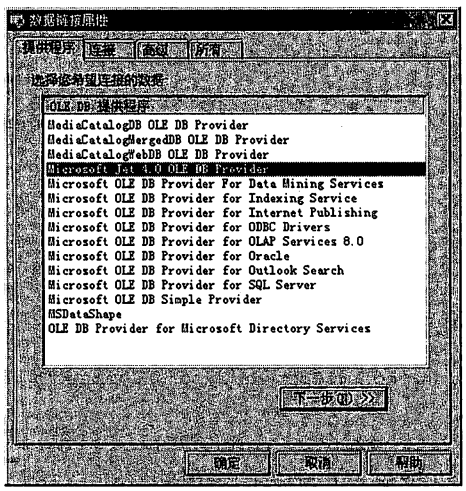


图 14-3　选择 OLE DB 驱动程序

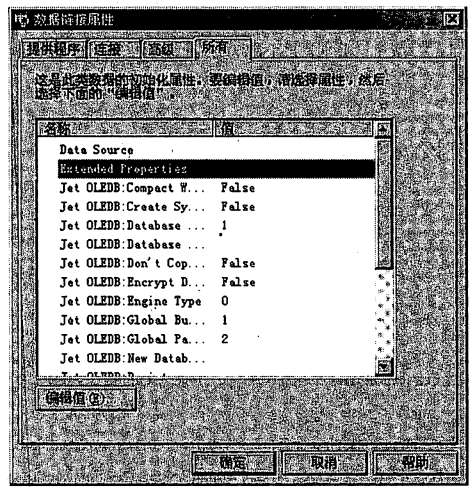


图 14-4　选择 Extended Properties

至此，组件成功连接到数据库，ConnectionString 属性设置完毕。

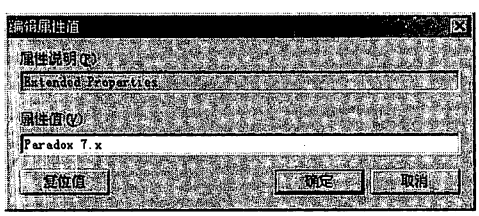


图 14-5　设置属性值为 Paradox 7.x

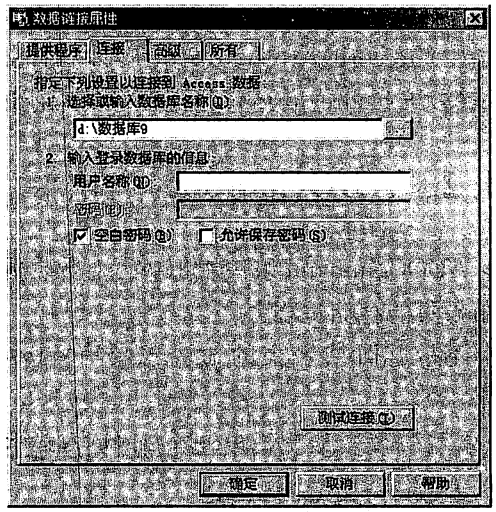


图 14-6　设置数据库名称

图 14-7　连接成功

5. CommandTimeout 属性

该属性为连接超时属性，默认值是 30 秒，即等待 30 秒还没有连接成功就放弃这个命

令，用户可以修改等待时间。

14.1.2 ADOConnection 组件的常用方法

1. Open 方法

该方法用于建立到数据库的物理连接。格式为：

```
procedure Open; overload;
procedure Open(const UserID: WideString; const Password: WideString); overload;
```

其中，UserID 和 Password 是数据库的用户名和登录数据库的口令。

2. Cancel 方法

该方法用于关闭与数据库的连接。格式为：

```
procedure Cancel;
```

3. Excute 方法

该方法用于执行指定的修改、查询等命令。格式为：

```
function Execute(const CommandText: WideString; const CommandType: TCommandType=
cmdText;
const ExecuteOptions: TExecuteOptions=[]): _RecordSet; overload;
procedure Execute(const CommandText: WideString; const CommandType: var
RecordsAffected: Integer; ExecuteOptions: TExecuteOptions=[eoExecuteNoRecords]);
overload;
```

说明：

(1) CommandText 是要执行的 SQL 命令、表名、存储过程。

(2) RecordsAffected 操作所影响的记录数。

(3) ExecuteOptions 指定命令的特征选择项。

14.2 ADOCommand 组件

ADOCommand 组件用于执行对数据源指定命令，主要用于执行 SQL 或一个不带返回结果集的存储过程。

不带返回结果集的命令主要有：Insert、Delete 和 Update，而 Select 通常要返回结果。

14.2.1 ADOCommand 组件的常用属性

1. CommandText 属性

该属性是用于表示命令的字符串，如 SQL 语句、表名或存储过程，默认值为空。如果要设置该属性，需要将 Prepared 设置为 True，并设置 Connection 属性。

2. CommandType 属性

该属性指定要执行的命令种类，它可以是表 14-2 中所列的任何一种。

表 14-2　CommandType 属性取值及其说明

取　值	含　义
cmdFile	保存数据集的文件名
cmdStoredProc	指定一个存储过程的名称
cmdTable	指定一个表的名称
cmdTableDirect	指定表的名称并返回所有的列
cmdText	将属性值作为命令或存储过程的文本
cmdUnknown	未知的命令类型，默认值

3. Connection 属性

该属性指定所要使用数据源连接组件的名称，即 ADOConnection 组件通过该属性 ADOCommand 组件可以与数据库连接起来。

14.2.2　ADOCommand 组件的常用方法

1. Cancel 方法

该方法用于取消上一步执行的命令。

2. Assign 方法

该方法用于把另外一个 ADOCommand 组件的所有属性复制到当前的 ADOCommand 组件中。格式如下：

```
procedure Assign(Source: TPersistent); override;
```

3. Execute 方法

该方法用于执行 ADOCommand 组件所包含的命令，返回结果是一个记录集。格式如下：

```
function Execute: _Recordset; overload;
function Execute(const Parameters: OleVariant): _Recordset; overload;
function Execute(var RecordsAffected: Integer; const Parameters: OleVariant): _
RecordSet; overload;
```

14.3　ADODataSet 组件

ADODataSet 组件是 ADO 组件中常见的 ADO 数据集组件，它可以从 ADO 数据库中读取一个或多个表，也可以通过 SQL 访问数据表。在使用 ADODataSet 组件之前，应建立它与数据库之间的关联，可以通过设置 ConnectionString 属性或 Connection 属性来指定一个 ADOConnection 组件。

14.3.1 ADODataSet 组件的常用属性

1. Connection 属性

该属性用于将 ADODataSet 组件关联到某个 ADOConnection 组件。

2. Active 属性

只有该属性为 True 时，应用程序才能对数据库进行读写操作。如果要更新数据集属性，需要先将 Active 属性设置为 False。

3. AutoCalcFields 属性

值为 True 时，允许触发 OnCalcFields 事件计算字段。该字段的值通过当前记录的一个或多个字段计算得到。

4. CommandText 属性

该属性是指定数据集中所包含的命令，可以是 SQL 语句、表名或存储过程。

14.3.2 ADODataSet 组件的常用方法

1. Cancel 方法

该方法用于取消上一步命令操作。

2. DeleteRecords 方法

该方法用于删除当前记录。格式为：

```
type TAffectRecords= (arCurrent, arFiltered, arAll, arAllChapters);
procedure DeleteRecords(AffectRecords: TAffectRecords=arAll);
```

其中，AffectRecords 用于指定要删除的记录，可以是如下的任何一种。

arCurrent：表示仅删除当前记录，是默认值。

arFiltered：删除满足过滤条件的记录。

arAll：删除全部记录。

arAllChapters：删除 ADO 连接数据部分的全部子集记录。

3. Locate 方法

该方法用于查询一条记录。格式为：

```
function Locate(const KeyFields: string; const KeyValues: Variant; Options:
TLocateOptions): Boolean; virtual;
```

其中，KeyFields 是索引的字段名。KeyValues 是要查找的值。Options 是查找规则选项，它的值可以有两种：loCaseInsensitive 表示查找不分大小写；loPartialKey 用于模糊查找。

4. Seek 方法

该方法将以当前数据集中的索引为依据查询并定位数据集指针。格式为：

```
type TSeekOption= (soFirstEQ, soLastEQ, soAfterEQ, soAfter, soBeforeEQ, soBefore);
```

```
function Seek (const KeyValues: Variant; SeekOption: TSeekOption = soFirstEQ):
Boolean;
```

其中，KeyValues 是要查找的关键值。SeekOption 是查找规则，有如下取值。

soFirstEQ：指针指向第一条满足条件的记录处。

soLastEQ：指针指向最后一条满足条件的记录处。

soAfterEQ：指针指向满足条件的记录的下一条记录；如果找不到则指向最近似的那条记录。

soAfter：指针指向满足条件的记录的下一条记录。

soBeforeEQ：指针指向满足条件的记录的上一条记录，如果找不到则指向最近似的那条记录。

soBefore：指针指向满足条件的记录的上一条记录。

5. MoveBy 方法

该方法用于移动记录指针，格式为：

```
function MoveBy(Distance: Integer): Integer;
```

其中，Distance 表示记录指针移动的条数。当 Distance 大于 0 时表示向下移动，当 Distance 小于 0 时表示向上移动。

6. First、Next、Last 和 Prior 方法

上述方法用于移动数据集中的记录指针，而 First、Next、Last 和 Prior 分别表示移到第一条记录、移到下一条记录、移动最后一条记录和移到前一条记录。

【例 14-1】 设计一个数据库应用程序。在 Delphi 中提供一些数据库表，它们在"...\Common Files\Borland Shared\Data"文件夹中。本程序可显示数据库 dbdemos. mdb 中的表文件 country。

设计步骤如下：

（1）新建应用程序，在界面添加组件 ADOConnection1，设置其属性 ConnectionString。双击该组件，出现对话框，如图 14-8 所示。

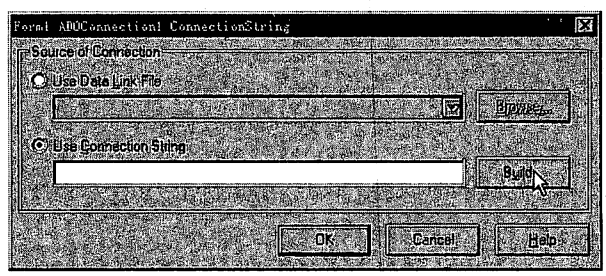


图 14-8　ConnectionString 对话框

（2）单击 Build 按钮，出现如图 14-9 所示对话框。在"提供程序"选项卡中选择 Microsoft Jet 4.0 OLE DB Provider，选择"连接"选项卡，单击"选择或者输入数据库名

称”编辑框右边的按钮，打开 Access 数据库，如图 14-10 所示。

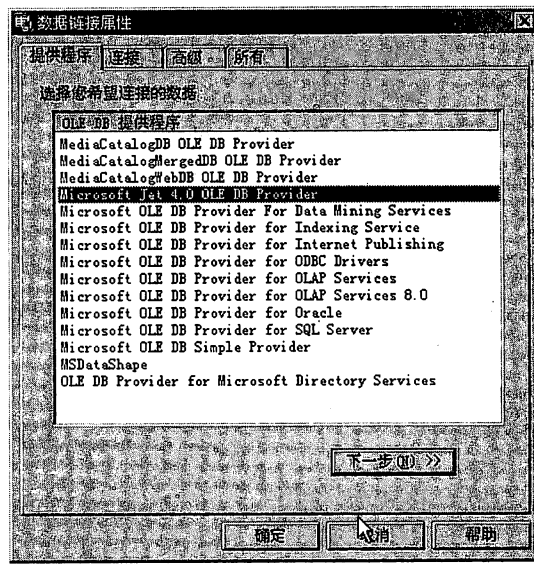


图 14-9　选择合适的提供程序

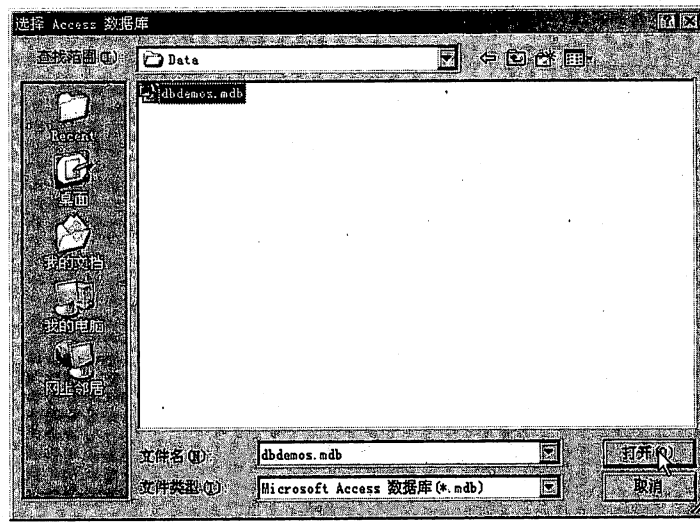


图 14-10　打开 Access 数据库

(3) 去掉“用户名称”编辑框的内容，单击“测试连接”，显示“测试连接成功”信息，如图 14-11 所示。

(4) 在界面上添加 ADOData 组件，设置其 Connection 属性为 ADOConnection1，设置其 CommandText 属性，在 SQL 编辑区输入“select * from country”，如图 14-12 所示。

(5) 添加 DataSource 组件，设置其 DataSet 属性为 ADODataSet1。

(6) 添加 DBGrid1，设置其 DataSource 为 DataSource1。此时设置 ADODataSet1 的 Active 属性为 True，即可在 DBGrid 中显示数据。

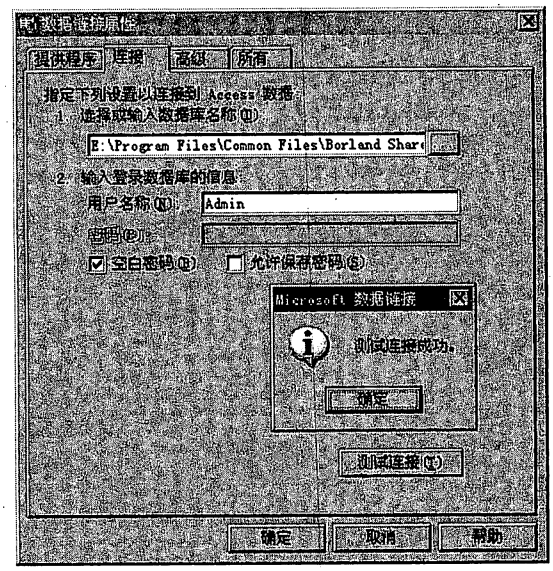


图 14-11　提示“测试连接成功”信息

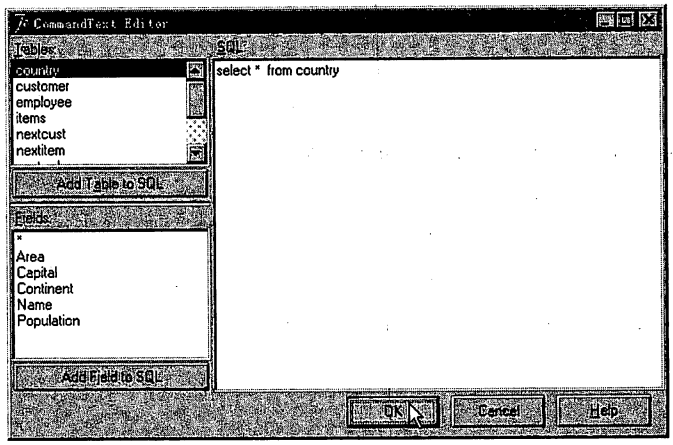


图 14-12　CommandText Editor 对话框

(7) 运行程序，结果如图 14-13 所示。

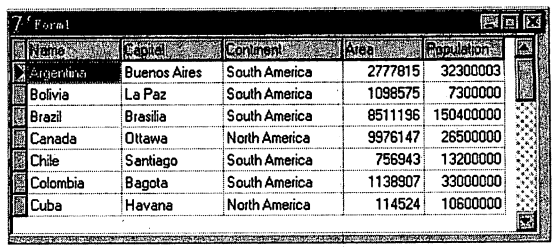


图 14-13　程序运行结果

说明：

(1) 可以在上面程序中添加组件 DBNavigator1，并设置其 DataSource 为 DataSource1，这样就可以使用 DBNavigator1 来控制数据库表文件了。

(2) 可以添加命令按钮，调用 ADODataSet 的方法来控制数据库表。例如，在如图 14-14中添加了 7 个按钮。

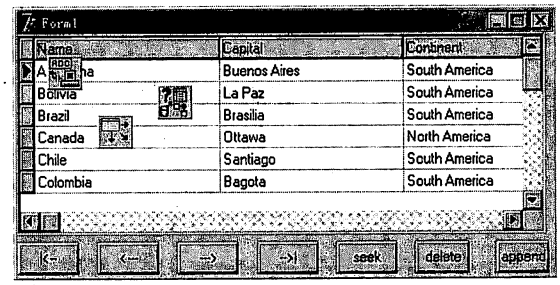


图 14-14 添加按钮后的界面

相应的代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 ADODataset1.First ;                          //显示第一条记录
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
 adodataset1.Prior;                           //显示前一条记录
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
 adodataset1.Next;                            //显示后一条记录
end;

procedure TForm1.Button4Click(Sender: TObject);
begin
 adodataset1.Last ;                           //显示最后一条记录
end;

procedure TForm1.Button5Click(Sender: TObject);
begin
 adodataset1.Locate('name','chile',[loCaseInsensitive]) ;
                                              //查询名称为"chile"的国家
end;

procedure TForm1.Button6Click(Sender: TObject);
begin
```

```
 adodataset1.Delete ;                          //删除记录
end;

procedure TForm1.Button7Click(Sender: TObject);
begin
  adodataset1.Append;                          //添加记录
end;
```

(3) 程序运行结果(省略)。

14.4　ADOTable 组件

ADOTable 组件可以通过 ADO 访问数据库中的单个表,也可以访问一个数据表中的所有数据记录和字段,还可以通过设置筛选条件来访问部分记录。

ADOTable 组件与 Table 组件非常相似,因此它们的很多属性和方法是一样的。ADOTable 组件最常用的属性和方法如下。

1. MasterSource 属性和 MasterFields 属性

在建立主从表关系时要用到 MasterSource 属性,用来设置主表的数据源。

MasterFields 属性,指定用于建立主从表主从关系的关联字段。

2. TableName 属性

该属性用于指定被操作的数据表的名字。

3. TableDirect 属性

该属性用于确定表是如何被引用的。值为 False 时 ADOTable 通过表访问数据库,值为 True 时 ADOTable 通过运行 SQL 命令访问数据库,默认值为 False。

4. Append 方法和 AppendRecord 方法

Append 方法用于在当前数据表中添加一条新记录,并将这条新记录设置为当前记录。AppendRecord 方法也是向当前数据表中添加一条新记录,不同的是它需要给新记录赋值,其格式为:

```
procedure AppendRecord(const Values: array of const);
```

其中,Values 是数组,且必须保证数组的维数和个数以及顺序与字段数匹配。

【例 14-2】 利用 ADOTable 组件设计一个数据库应用程序。在 Delphi 中提供一些数据库表,它们在"...\Common Files\Borland Shared\Data"文件夹中。本程序可显示数据库 dbdemos.mdb 中的表文件 country。

设计步骤如下:

(1) 新建应用程序,在窗体上添加组件 ADOConnection1、ADOTable1、DataSource1 和 DBGrid1。设置属性如表 14-3 所示。

表 14-3 组件属性设置

组　件	属　性	属 性 值
ADOConnection1	ConnectionString	设置该值，使之关联到 dbdemos. mdb
	LoginPrompt	False
	Connected	True
ADOTable1	Connection	ADOConnection1
	TableName1	country(dbdemos. mdb 中的一个表)
	Active	True
DataSource1	DataSet	ADOTable1
DBGrid1	DataSource	DataSource1

(2) 在窗体上添加若干命令按钮，用于操纵数据库表。界面如图 14-15 所示。

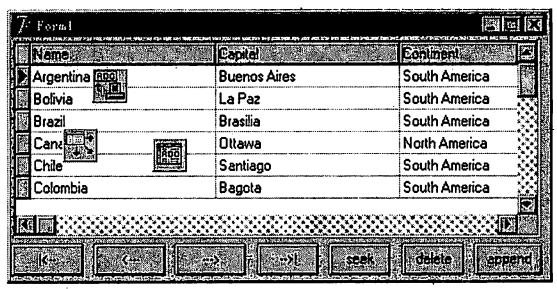


图 14-15 程序界面

程序代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 adotable1.First;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
 adotable1.Prior;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
 adotable1.Next ;
end;
```

```
procedure TForm1.Button4Click(Sender: TObject);
begin
 adotable1.Last;
end;

procedure TForm1.Button5Click(Sender: TObject);
begin
 adotable1.Locate('name','chile',[]);      //country 表中国名 Name 已经设置索引
end;

procedure TForm1.Button6Click(Sender: TObject);
begin
 adotable1.Delete ;
end;
procedure TForm1.Button7Click(Sender: TObject);
begin
 adotable1.Append;
end;
```

(3) 程序运行结果(省略)。

14.5　ADOQuery 组件

ADOQuery 组件与 Query 组件很类似,可以使用 SQL 语句在一个和多个数据库之间选择字段,而且可以增加或删除数据。在使用该组件的时候,必须先把它连接到数据库,可以使用 ConnectionString 属性来连接,也可以通过设置其 Connection 属性为某个 ADOConnection 组件来连接。

1. SQL 属性

该属性用于指定利用 ADOQuery 组件来执行的 SQL 语句。SQL 语句不仅可以是 Select 语句,还可以是 Delete、Insert、Update 等语句。

2. Open 方法和 ExecSQL 方法

当 SQL 属性是 Select 语句时,使用 Open 方法执行 SQL 语句,否则使用 ExecSQL 方法执行 SQL 语句。

【例 14-3】 使用 ADOQuery 组件操纵数据库表 dbdemos. mdb。

设计步骤如下:

(1) 新建应用程序,在窗体上添加组件 ADOQuery1、DataSource1、DBGrid1、ADOConnection1、Edit1 和 Button1。

(2) 设置属性,如表 14-4 所示。

表 14-4 各组件的属性设置

组　件	属　性	属　性　值
ADOConnection1	ConnectionString	设置该值，使之关联到 dbdemos. mdb
	LoginPrompt	False
	Connected	True
ADOQuery1	Connection	ADOConnection1
	Active	True
DataSource1	DataSet	ADOQuery1
DBGrid1	DataSource	DataSource1

(3) 调整组件位置，界面如图 14-16 所示。

(4) 编写代码如下：

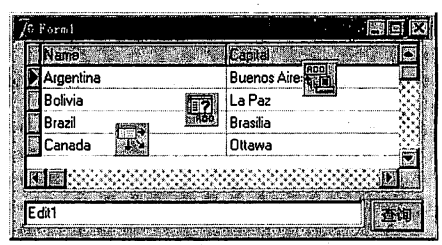


图 14-16 程序界面

```
procedure TForm1. Button1Click (Sender:
TObject);

 var st1,st2:string;

begin
 st1:=trim(edit1.Text);
 st2:=copy(st1,0,6);
 if st2='select'                          //前 6 个字符是否为'select'
 then begin
      adoquery1.SQL.Clear;
      adoquery1.SQL.Add(edit1.Text);
      adoquery1.Open;                     //若是 select 语句则调用 open 方法
     end
 else begin
      adoquery1.SQL.Clear;
      adoquery1.SQL.Add(st1);
      adoquery1.ExecSQL;                  //不是 select 语句则调用 ExecSQL 方法
     end;
end;
```

说明：本题还可以不要 ADOConnection 组件，其他组件保持不变，属性设置如表 14-5 所示，程序代码和运行结果与原程序完全一样。

表 14-5 组件的属性设置

组　件	属　性	属　性　值
ADOQuery1	ConnectionString	设置该值，使之关联到 dbdemos. mdb
	Active	True
DataSource1	DataSet	ADOQuery1
DBGrid1	DataSource	DataSource1

14.6　使用 ADO 开发数据库综合实例

使用 ADO 组件编写教师信息管理系统。该系统要求显示教师信息，包括基本信息和教学信息。要求可以添加、修改、浏览、查询、删除、更新这两个表的信息。

1. 建立数据库

使用 Access 建立数据库“教师信息.mdb”，该数据库含有“基本表”和“教学表”，这两个表的结构如图 14-17 所示。向表中输入若干条记录，将数据库“教师信息.mdb”保存在事先建立的文件夹“D:\Delphi 程序\ADO 综合”中。

基本表：表

字段名称	数据类型
工号	文本
姓名	文本
性别	文本
电话	文本
地址	文本
婚否	是/否
备注	文本

教学表：表

字段名称	数据类型
工号	文本
姓名	文本
专业	文本
职称	文本
工龄	数字
工资	数字
奖金	数字

图 14-17　教师信息.mdb 中两个表的表结构

2. 设计程序界面

新建应用程序，在窗体 Form1 中添加组件 PageControl1 和 Panel1，将 PageControl1 的 Align 属性设置为 Altop，将 Panel1 的 Align 属性设置为 AlClient，并调整好这两个组件的高度。

为 PageControl1 新建两页，这两页的 Caption 分别设置为“基本表”和“教学表”。在“基本表”页面中添加 DBGrid1 和 DBNavigator1，在“教学表”页面中添加 DBGrid2 和 DBNavigator2。

在 Panel1 中添加 DBGrid3、Edit1 和 Button1。在 Form1 中添加组件 ADOConnection1、ADOTable1 和 DataSource1，这 3 个组件用于显示基本表。

在窗体 Form1 中添加组件 ADOTable2 和 DataSource2，这两个组件以及 ADOConnection1 用于显示教学表。在窗体 Form1 中添加组件 ADOQuery1 和 DataSource3，这两个组件以及 ADOConnection1 用于显示数据库的查询信息。

程序界面如图 14-18 所示。

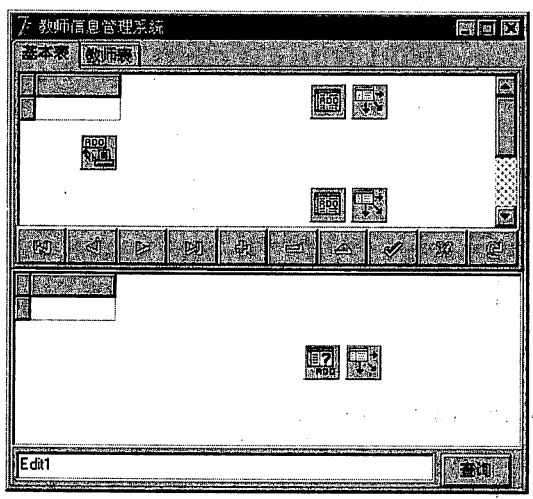


图 14-18　程序界面

3. 属性设置

设置窗体中各个组件的属性，如表 14-6 所示。

表 14-6 各组件的属性设置

对 象	属 性	属 性 值	说 明
ADOConnection1	ConnectionString	省略	连接到教师信息.mdb
ADOTable1	Connection	ADOConnection1	连接到基本表
	TableName	基本表	
	Active	True	
DataSource1	DataSet	ADOTable1	
DBGrid1	DataSource	DataSource1	用于显示基本表
DBNavigator1	DataSource	DataSource1	用于操作基本表
ADOTable2	Connection	ADOConnection1	连接到教学表
	TableName	教学表	
	Active	True	
DataSource2	DataSet	ADOTable2	
DBGrid2	DataSource	DataSource2	用于显示教学表
DBNavigator2	DataSource	DataSource2	用于操作教学表
ADOQuery1	ADOConnection	ADOConnection1	用于查询教师信息.mdb
DataSource3	DataSet	ADOQuery1	
DBGrid3	DataSource	DataSource3	
TabSheet1	Caption	基本表	
TabSheet2	Caption	教学表	
Edit1	Text	设置为空	
Button1	Caption	查询	实现查询功能

4. 编写代码

编写“查询”按钮的代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 adoquery1.Close;
 adoquery1.SQL.Clear;
```

```
 adoquery1.SQL.Add(edit1.Text);
 adoquery1.Open ;
end;
```

至此，教师信息管理系统编写完成，程序运行正常。

5. 说明

为了让程序可以在各个文件夹中都能够正常运行，可以将 ADOConnection1 的属性 ConnectionString 在 Form1 的 OnCreate 事件过程中动态设置。程序如下：

```
procedure TForm1.FormCreate(Sender: TObject);

 var curdir:string;

begin
 getdir(0,curdir);
 adoconnection1.ConnectionString:='Provider=Microsoft.Jet.OLEDB.4.0;Data
   Source='+curdir+'\教师信息.mdb;Persist Security Info=False';
  AdoConnection1.Open;
 ADOTable1.TableName:='基本表';
 ADOTable1.TableName:='教学表';
 ADOTable1.Open;
 ADOTable2.Open;
end;
```

值得注意的是，如果对 ADOConnection1 的 ConnectionString 属性进行动态设置，则 ADOTable1 和 ADOTable2 的 Active 属性都要设置为 False，且 ADOTable1 和 ADOTable2 的 TableName 也要在 FormCreate 事件过程动态设置，然后再执行 ADOTable1 和 ADOTable2 的 Open 方法。

14.7　小结

本章讲述 ADO 数据库应用程序开发方法，主要内容如下。

- ADOConnection 组件的用法，重点讲述 ADOConnection 组件的属性。
- ADOCommand 组件的用法。
- ADODataSet 组件的用法。
- ADOTable 组件的用法。
- ADOQuery 组件的用法。
- 使用 ADO 组件开发教师信息管理系统。读者在编写数据库管理系统或查询系统时，应该注意如何让编写的系统在各个不同的路径下正常运行。

本章重点是 ADOConnection 组件、ADOTable 组件和 ADOQuery 组件，并务必学会使用 ADO 组件编写数据库应用程序。

习题

1. Delphi 通过把 ADO 的对象封装在相应的组件中来实现对 ADO 的支持，通常可以使用_____组件来建立与物理数据库的连接，其他组件可以通过访问该组件来连接物理数据库。

A. ADOConnection　　B. ADOTable

C. ADOCommand　　D. ADOQuery

2. 在开发较为复杂的 ADO 数据库应用程序时，其他 ADO 组件（如 ADOTable 组件）最好还是通过 ADOConnection 组件来连接数据库，此时，应把它的_____属性设置为 ADOConnection 组件名。

A. ConnectionTimeOut　　B. Connected

C. ConnectionString　　D. Connection

3. 下面_____组件可以执行 SQL 命令。

A. ADOConnection　　B. ADOTable

C. ADOCommand　　D. ADOQuery

4. Table 组件的_____属性可以取得被操作表的表名。

A. TableName　　B. TableDirect

C. CommandType　　D. CommandText

5. 编写一个数据库应用程序，该数据库应用程序通过执行 SQL 访问两张数据库表 cj. db 和 cj1. db（这两张表的结构与数据和“D:\delphi 程序\ch13\eg13-6”中的两张表一样）。

6. 编写 ADO 应用程序，输入学号，显示该学生的成绩信息，数据库表与第 5 题相同。

7. 编写一个 ADO 数据库应用程序，要求该程序访问“... Program Files\Common Files\Borland Shared\Data”下的 biolife. db。要求显示所有的字段。

第 15 章

报表设计

RAVE(Report Authoring Visual Enviroment)是强大的报表设计工具,它是Nevrona Designs公司的产品,有三个版本:标准版、开发版和Borland版。在Delphi 7中提供的是Borland版RAVE 5.0,RAVE 5.0提供了很多强大的组件,使用这些组件可以极为方便地开发出应用程序报表。RAVE 5.0报表组件位于RAVE组件选项卡中,本章将介绍其中最重要的几个组件。

15.1 RvProject组件

RvProject组件是访问可视化报表的手段,它是Rave报表最重要的组件。其主要属性和方法如下。

(1) Projectfile属性:指定报表项目文件名。当报表项目文件打开的时候,RvProject必须指向一个有效的报表项目文件名。报表项目文件的扩展名是rav。

(2) Close方法:在修改报表属性的时候,一般需要先关闭RvProject,Close方法用于关闭RvProject,并从内存释放报表。

(3) Open方法:启动报表项目,在执行报表之前需要先启动报表。

(4) Save方法:保存报表项目文件名。

(5) Execute方法:执行当前选定的报表项目文件。

15.2 RvDataSetConnection组件

该组件提供数据源,设置其DataSet属性可以连接到数据源。利用Rave新建一个Direct Data View(视图)可以将RvProject与RvDataSetConnection组件关联起来。这在后面的例题中会详细讲述。

该组件访问DataSet组件,提供Rave报表与数据源的连接。主要属性有DataSet属性,该属性指定一个DataSet,连接到数据源。

15.3 DataText、DataMemo 和 DataBitMap 组件

这些组件用于显示表的不同的字段。例如，DataText 和 DataMemo 可以用来显示字符型字段，而 DataBitMap 则可以用来显示图片类字段。上述组件的主要属性如下。

(1) DataView：指定其相应的数据连接视图。

(2) DataField：指定显示的字段。

【例 15-1】 在应用程序中使用报表。本程序将以"...\Common Files\Borland Shared\Data"文件夹中的 country.db 为例讲述报表的打印。

详细的设计步骤如下：

(1) 新建文件夹"d:\ch15"，新建 Delphi 应用程序 application。

(2) 新建数据模块，选择菜单 File | New | Data Module。将该模块改名(修改模块的 Name 属性)为 dm。

(3) 在数据模块中添加组件 DataSource1 和 Table1，如图 15-1 所示。

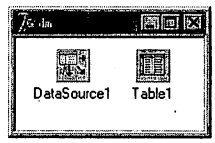


图 15-1 dm 数据模块中的组件

(4) 设置上述组件的属性，如表 15-1 所示。

表 15-1 数据模块中组件属性设置

组 件	属 性	属 性 值
Table1	DatabaseName	DBDemos
	TableName	country.db
	Active	True
DataSource	DataSet	Table1

(5) 在 Form1 中添加组件 DBGrid1、Button1、RvProject1 和 RvDataSetConnection1，设置属性如表 15-2 所示。

表 15-2 窗体中各个组件的属性设置

组 件	属 性	属 性 值
RvDataSetConnection1	DataSet	dm.Table1
DBGrid1	DataSet	dm.DataSource1

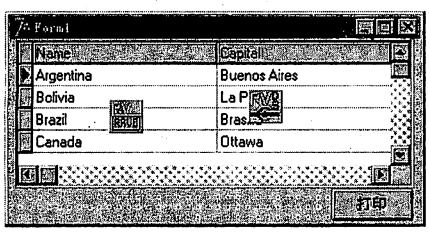


图 15-2 设置属性后的程序界面

设置属性后的界面如图 15-2 所示。

(6) 双击 RvProject1 组件启动 Rave 5，界面如图 15-3 所示。

该界面是可视化报表生成器环境。最上面是菜单项；菜单项下面是工具栏，工具栏保存着菜单中最常用命令；工具栏右边是组件选项卡；下面的

左边是组件的属性设置对话框，用法与 Delphi 7 相同；中间区域是报表设计区域；右边是报表对象导航器。

(7) 在 Rave Reports 5.0 环境中新建数据对象，选择菜单 File|New Data Object，出现“数据连接”对话框，选择 Direct Data View，单击 Next 按钮，如图 15-4 所示。

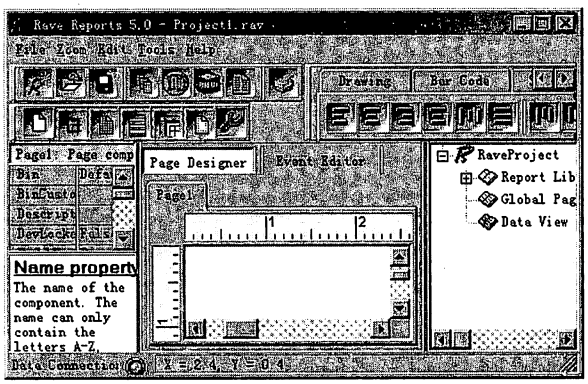


图 15-3　Rave Reports 5.0 界面

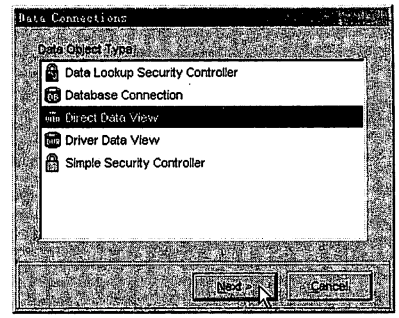


图 15-4　“数据连接”对话框

(8) 在出现的对话框中选择 RvDataSetConnection1(DT)，并单击 Finish 按钮。此时已经将报表连接到数据源，因为 RvConnection1 此前已经连接到数据源(dm. table1)，如图 15-5 所示。

(9) 利用报表向导设计报表(还可以将向导设计好的报表，利用手工方式加以修改)。选择菜单 Tool|Report Wizards|Simple Table，界面如图 15-6 所示。选择 DataView1，单击 Next 按钮。

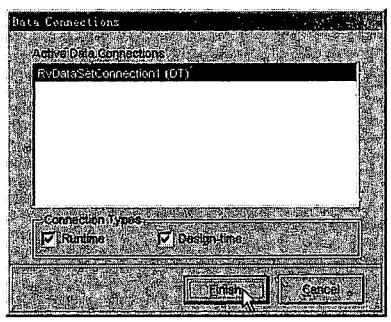


图 15-5　连接到数据源

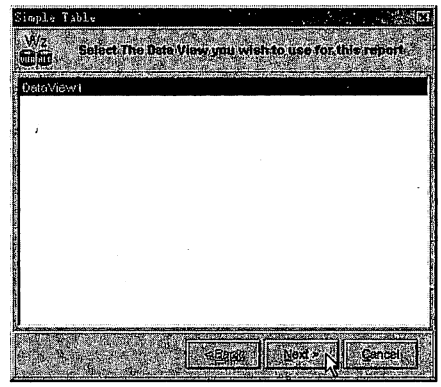


图 15-6　报表设计向导(选择数据视图)

(10) 出现如图 15-7 所示界面，在此选择要打印的字段。单击 All 按钮选中所有字段，再单击 Next 按钮。

(11) 选择打印顺序，如图 15-8 所示，直接单击 Next 按钮。

(12) 打印边框和打印标题设置，如图 15-9 所示。在此不做修改，可以在后面再做修改。单击 Next 按钮。

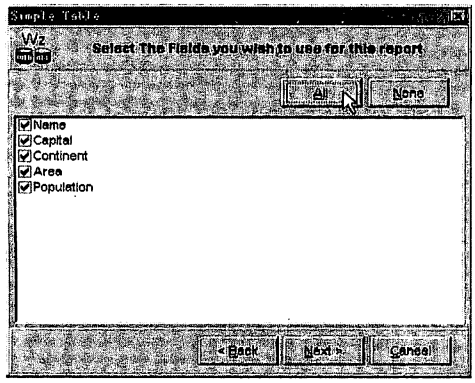


图 15-7 选择字段

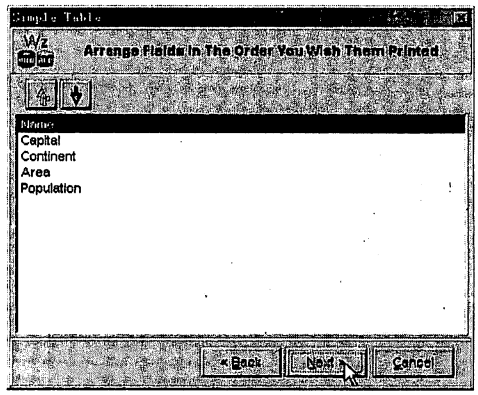


图 15-8 选择打印顺序

(13) 在图 15-9 界面中单击 Next 按钮后出现如图 15-10 的界面，此处可以设置标题、标签等字体。此处不做修改，直接单击 Generate 按钮生成报表。

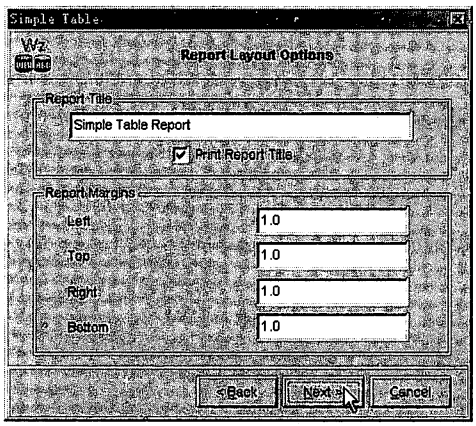


图 15-9 报表边框与标题设置

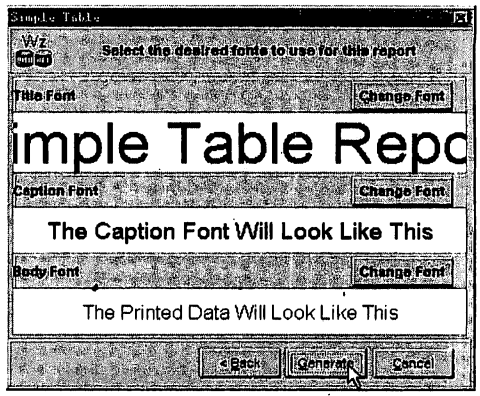


图 15-10 修改字体界面

(14) 向导生成的报表如图 15-11 所示。

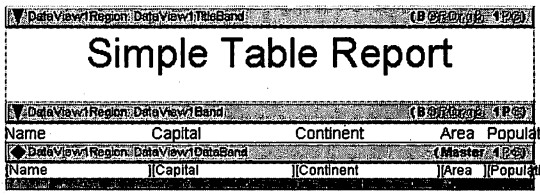


图 15-11 报表向导生成的报表

(15) 调整 Text、DataText 等组件的位置，并修改 TitleText 的 Text 属性为“美洲国家”，修改字体大小，调整标题带高度。调整好的报表界面如图 15-12 所示。

(16) 保存所有文件。在 Delphi 中单击“保存所有”按钮，将这些文件保存到 ch15 文件夹中。在 Rave Reports 5 中选择“保存按钮”，保存报表文件到 ch15 文件夹下，并

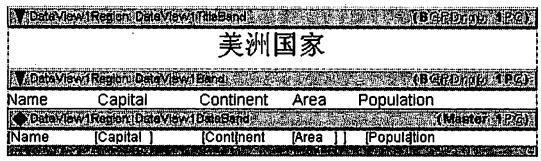


图 15-12　修改后的报表

命名为 Project1.rav。

(17) 为“打印”按钮编写代码，打印所设计的报表。代码如下：

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 //关闭当前报表项目文件
 RvProject1.Close;
 //载入"RaveDemo.rav"报表
 RvProject1.ProjectFile:= ExtractFilePath (Application.ExeName) + ' Project1.
rav';
 //打开选定的报表项目文件
 RvProject1.Open;
 //启动报表
 RvProject1.Execute;
end;
```

(18) 运行程序，结果如图 15-13 所示。单击“打印”按钮，出现如图 15-14 所示界面。单击 OK 按钮，预览效果如图 15-15 所示，单击图标开始打印。

图 15-13　单击“打印”按钮

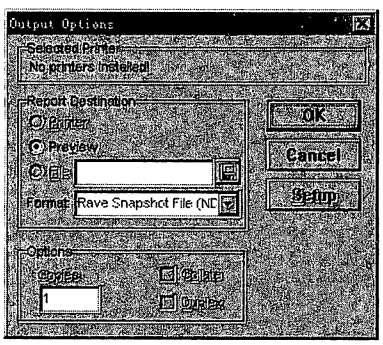


图 15-14　单击 OK 按钮

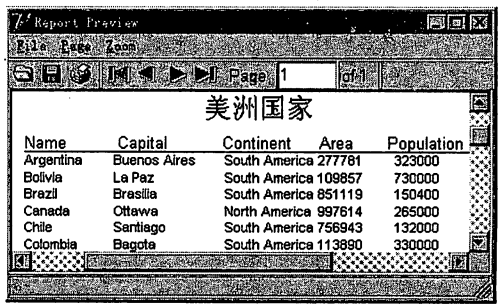


图 15-15　预览效果

15.4　小结

本章讲述了 Rave 5 报表，具体讲述了 RvProject 组件及 RvDataSetConnection 组件、DataText 组件、DataMemo 组件和 DataBitMap 组件，并通过具体实例讲述了如何制作

Rave 5 报表。教师和读者可以根据自己的实际情况自行决定是否讲授或者学习本章。

习题

1. 使用 Rave 组件制作报表，打印“... Program Files\Common Files\Borland Shared \Data”下的 biolife. db。要求：

(1) 打印所有记录。

(2) 打印指定的某条记录。

2. 请从 http://www. fast-report. com 下载 FastReport 报表组件，并安装该组件(多个组件)，尝试使用 FastReport 组件设计报表。

第 16 章

网 络 编 程

计算机网络可以将地理位置不同的多台计算机通过通信线路和设备连接起来，并在软件的支持下实现互相通信和资源共享。

16.1 使用 Delphi 网络组件

Delphi 7 提供了十分丰富的网络组件，利用这些组件可以十分方便地编制网络应用程序，其功能非常强大，无须用户编写太多代码，即可完成一些基本的网络功能。这些网络组件存放在 Delphi 7 的 Internet 组件选项卡上，如图 16-1 所示。

图 16-1 Internet 组件选项卡上的网络组件

其他组件选项卡上也提供了一些网络组件，但这里不作介绍。

16.1.1 TCPServer 组件和 TCPClient 组件

TCPServer 组件()建立 TCP/IP 的服务器端，TCPClient 组件()建立 TCP/IP 的客户端。

1. TCPServer 的主要属性

(1) LocalPort 属性：指定 TCP 端口号。

(2) LocalHost 属性：指定服务器名称。

2. TCPServer 的主要事件

(1) OnClientConnect 事件：与客户机连接成功后发生该事件。

(2) OnClientDisconnect 事件：与客户机结束连接后发生该事件。

(3) OnClientRead 事件：从客户机读数据时发生该事件。

(4) OnClientWrite 事件：向客户机写数据时发生该事件。

3. TCPClient 的主要属性

(1) RemotePort 属性：与 TCPServer 组件的 LocalPort 属性相对应。

(2) RemoteHost 属性：与 TCPServer 组件的 LocalHost 属性相对应。

4. TCPClient 的主要事件

(1) OnLookUp 事件：客户端企图和服务器连接时发生该事件。

(2) OnConnect 事件：客户端和服务器成功连接后发生该事件。

(3) OnDisconnect 事件：与服务器结束连接后发生该事件。

(4) OnRead 事件：当从 TCPClient 读数据时发生该事件。

(5) OnWrite 事件：当向 TCPClient 写数据时发生该事件。

【例 16-1】 聊天程序，利用 TCPServer 和 TCPClient 组件编写简单的聊天程序。

设计步骤如下：

(1) 在窗体上添加组件 Panel1，设置 Align 属性为 AlBottom，在 Panel1 中添加多行文本框 Memo1 并改名为 MemSend。在 Panel1 中添加 Panel2，设置 Panel2 的 Align 属性为 Alright，在 Panel2 中添加按钮 Button1，并改名为 btnsend。

(2) 在窗体空白处添加 Panel3，设置 Panel3 的 Align 属性为 AlTop，在 Panel3 中添加标签 Label1、Label2 和 Label3，设置 Caption 分别为 RemoteHost、RemotePort 和 Local Port。添加三个编辑框并改名为 edtremotehost、edtremoteport 和 edtlocalport。

(3) 在窗体空白处添加编辑框，并改名 edtrecv，设置 Align 为 AlClient。

聊天程序界面如图 16-2 所示。

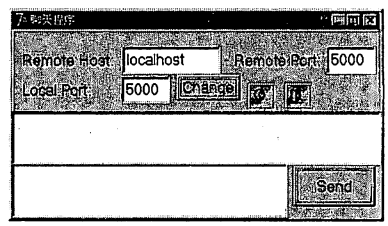


图 16-2 聊天程序界面

(4) 编写代码如下：

```
unit Chatmain;

interface

uses
  Classes, QControls, QStdCtrls, QExtCtrls, QButtons, QForms, Sockets;

type
  TForm1=class(TForm)
    memRecv: TMemo;
    Panel1: TPanel;
    memSend: TMemo;
    Panel2: TPanel;
    btnSend: TButton;
    Panel3: TPanel;
    Label1: TLabel;
    edtRemoteHost: TEdit;
    Label2: TLabel;
    edtRemotePort: TEdit;
    Label3: TLabel;
    edtLocalPort: TEdit;
    btnActivateServer: TButton;
    TcpClient1: TTcpClient;
    TcpServer1: TTcpServer;
    procedure btnSendClick(Sender: TObject);
```

```
    procedure TcpServer1Accept(sender: TObject;
      ClientSocket: TCustomIpClient);
    procedure btnActivateServerClick(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

  TClientDataThread=class(TThread)                    //线程类声明

  private
  public
    ListBuffer :TStringList;                          //发送的信息
    TargetList :TStrings;
    procedure synchAddDataToControl;                  //添加信息
    constructor Create(CreateSuspended: Boolean);     //构造器
    procedure Execute; override;
    procedure Terminate;                              //终止线程
  end;

var
  Form1: TForm1;

implementation
{$R *.xfm}

//------------ TClientDataThread impl ------------------------------------
constructor TClientDataThread.Create(CreateSuspended: Boolean);
begin
  inherited Create(CreateSuspended);
  FreeOnTerminate :=true;
  ListBuffer :=TStringList.Create;
end;

procedure TClientDataThread.Terminate;
begin
  ListBuffer.Free;
  inherited;
end;

procedure TClientDataThread.Execute;                  //执行线程
begin
  Synchronize(synchAddDataToControl);
end;

procedure TClientDataThread.synchAddDataToControl;
begin
```

```
 TargetList.AddStrings(ListBuffer);                    //添加信息
end;
//------------ end TClientDataThread impl ----------------------------------

procedure TForm1.btnActivateServerClick(Sender: TObject);
begin
  TcpServer1.LocalPort :=edtLocalPort.Text;
  TcpServer1.Active :=True;
end;

procedure TForm1.btnSendClick(Sender: TObject);       //客户端程序

var
  I: Integer;

begin
  TcpClient1.RemoteHost :=edtRemoteHost.Text;
  TcpClient1.RemotePort :=edtRemotePort.Text;
  Try                                                 //尝试与服务器连接
    if TcpClient1.Connect then                        //连接成功
      for I :=0 to memSend.Lines.Count-1 do
      TcpClient1.Sendln(memSend.Lines[I]);            //发送信息
  finally
    TcpClient1.Disconnect;                            //断开连接
  end;
end;

procedure TForm1.TcpServer1Accept(sender: TObject;
  ClientSocket: TCustomIpClient);

var
  s: string; DataThread: TClientDataThread;

begin
  DataThread:=TClientDataThread.Create(true);         //生成线程
  DataThread.TargetList :=memRecv.lines;              //取得客户端内容
  DataThread.ListBuffer.Add('Remote Host: ' +
ClientSocket.LookupHostName(ClientSocket.RemoteHost) +
   ' (' +ClientSocket.RemoteHost +')');               //取得信息
  s :=ClientSocket.Receiveln;
  while s<>'' do
  begin
    DataThread.ListBuffer.Add(s);
    s :=ClientSocket.Receiveln;
  end;
  DataThread.Resume;
end;
end.
```

(5) 运行结果如图 16-3 所示。

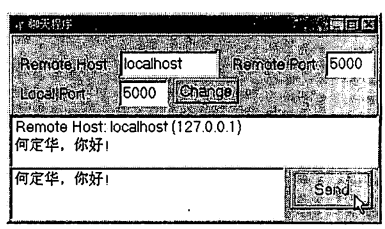


图 16-3 运行结果

说明：

(1) 本程序将客户机代码和服务器端代码写在一个应用程序中。它也可以做成两个应用程序，分成客户端程序和服务器端程序，从而在两台机器上运行，其中“procedure TForm1. btnSendClick(Sender: TObject);”是客户端程序，其他过程是服务器端程序。

(2) 此程序在“...\Borland\Delphi7\Demos\Internet\NetChat”文件夹中。

16.1.2　WebBrowser 组件

对于浏览器，读者一定不会陌生，微软公司的 IE 是大家最为熟悉的浏览器。建立浏览器有很多途径，可以从底层协议开始做，也可以使用已经封装好的现存的组件来做。WebBrowser 就是这样一个组件，它位于 Internet 组件选项卡上。

1. Navigate 方法

该方法用于指定网页的 URL。它有多种重载格式：

```
procedure Navigate(const URL: WideString); overload;
procedure Navigate(const URL: WideString; var Flags: OleVariant); overload;
procedure Navigate (const URL: WideString; var Flags: OleVar iant; var TargetFrame-
Name: OleVariant); overload;
procedure Navigate (const URL: WideString; var Flags: OleVariant; var TargetFrame-
Name: OleVariant; var PostData: OleVariant); overload;
procedure Navigate (const URL: WideString; var Flags: OleVariant; var TargetFrame-
Name: OleVariant; var PostData: OleVariant; var Headers: OleVariant); overload;
```

这里只介绍第一种格式，参数 URL 为统一资源定位符。即网页所对应的网址。

2. GoBack 方法

该方法用于返回最后一次历史记录 URL。格式为：

```
procedure GoBack;
```

3. GoForward 方法

该方法用于在历史记录中前进。格式为：

```
procedure GoForward;
```

4. GoHome 方法

该方法返回主页或起始页，主页或起始页在注册表的 HKEY_CURRENT_USER\Software\Microsoft\Internet Explorer\Main 处。该方法的格式为：

```
procedure GoHome;
```

5. LocationURL 属性

该属性表示当前显示页的资源分配符，它可以是网络上的一个网址或者本地机器上的一个路径。格式为：

```
property LocationURL: WideString;
```

6. OnDownloadBegin 事件

该事件发生在 WebBrowser 开始下载并显示网页时。格式为：

```
property OnDownloadBegin: TNotifyEvent;
```

7. OnDownloadComplete 事件

该事件发生在 WebBrowser 下载完毕网页时。格式为：

```
property OnDownloadComplete: TNotifyEvent;
```

【例 16-2】 利用 WebBrowser 组件设计一个简易的浏览器程序。

设计步骤如下：

(1) 添加组件 Panel1，设置 Panel1 的 Align 属性为 AlTop，在 Panel1 上添加组件 SpeedButton1、SpeedButton2、SpeedButton3、SpeedButton4、SpeedButton5、Edit1 和 SpeedButton6。SpeedButton 按钮的 Caption 分别为“后退”、“前进”、“主页”、“停止”、“刷新”和“出发”。

(2) 添加 StatusBar1，设置其 Align 为 AlBottom。

(3) 添加组件 WebBrowser1，设置其 Align 为 AlClient。

(4) 调整上述组件的大小和位置，界面如图 16-4 所示。

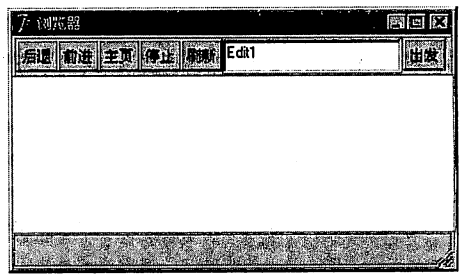


图 16-4 浏览器程序界面

(5) 编写代码如下：

```
procedure TForm1.SpeedButton1Click(Sender: TObject);
begin
 try
  WebBrowser1.GoBack
 except
  showmessage('已经到达最早历史记录');
  exit;
 end;
end;

procedure TForm1.SpeedButton2Click(Sender: TObject);
begin
 try
  WebBrowser1.GoForward;
 except
  showmessage('已经到达最后历史记录');
  exit;
  end;
end;

procedure TForm1.SpeedButton3Click(Sender: TObject);
```

```
begin
 WebBrowser1.GoHome;
end;

procedure TForm1.SpeedButton4Click(Sender: TObject);
begin
 WebBrowser1.Stop;
end;

procedure TForm1.SpeedButton5Click(Sender: TObject);
begin
 WebBrowser1.Refresh;
end;

procedure TForm1.SpeedButton6Click(Sender: TObject);
begin
 WebBrowser1.Navigate(edit1.Text);
end;

procedure TForm1.WebBrowser1DownloadBegin(Sender: TObject);
begin
 statusbar1.SimpleText:='正在打开'+edit1.Text;
end;

procedure TForm1.WebBrowser1DownloadComplete(Sender: TObject);
begin
 statusbar1.SimpleText:='已经连接到'+edit1.Text;
end;
```

(6) 运行结果如图 16-5 所示。

图 16-5　程序运行结果

16.2　小结

本章比较简单地讲述了 Delphi 网络编程，具体介绍了最常用的组件 TCPServer 组件、TCPClient 组件以及如何使用这两个组件编写聊天程序。本章还讲述了 WebBrowser 组件以及使用该组件编写浏览器。通过本章的学习，读者应该学会使用上述组件编写网

络应用程序。

习题

1. 使用 WebBrowser 制作一个浏览器，要求界面尽量美观，具备一些常用功能。

2. 使用 TCPServer 组件和 TCPClient 组件制作聊天程序，要求界面美观，服务器端程序和客户端程序分开编写(共两个程序)。

计算机精品学习资料大放送

软考官方指定教材及同步辅导书下载 | 软考历年真是解析与答案

软考视频 | 考试机构 | 考试时间安排

Java 一览无余：Java 视频教程 | Java SE | Java EE

.Net 技术精品资料下载汇总：ASP.NET 篇

.Net 技术精品资料下载汇总：C#语言篇

.Net 技术精品资料下载汇总：VB.NET 篇

撼世出击：C/C++编程语言学习资料尽收眼底 电子书+视频教程

Visual C++(VC/MFC)学习电子书及开发工具下载

Perl/CGI 脚本语言编程学习资源下载地址大全

Python 语言编程学习资料(电子书+视频教程)下载汇总

最新最全 Ruby、Ruby on Rails 精品电子书等学习资料下载

数据库精品学习资源汇总：MySQL 篇 | SQL Server 篇 | Oracle 篇

最强 HTML/xHTML、CSS 精品学习资料下载汇总

最新 JavaScript、Ajax 典藏级学习资料下载分类汇总

网络最强 PHP 开发工具+电子书+视频教程等资料下载汇总

UML 学习电子资下载汇总 软件设计与开发人员必备

经典 LinuxCBT 视频教程系列 Linux 快速学习视频教程一帖通

天罗地网：精品 Linux 学习资料大收集(电子书+视频教程) Linux 参考资源大系

Linux 系统管理员必备参考资料下载汇总

Linux shell、内核及系统编程精品资料下载汇总

UNIX 操作系统精品学习资料<电子书+视频>分类总汇

FreeBSD/OpenBSD/NetBSD 精品学习资源索引 含书籍+视频

Solaris/OpenSolaris 电子书、视频等精华资料下载索引

附录

标准过程与函数

Delphi 在 System 等多个单元中定义了一些标准函数和过程，现将其中一部分最常用的函数与过程列举出来，供读者参考。

表 1　System 单元中的函数与过程

分　类	函　数　名	参数类型	结果类型	说　　明
算术函数	绝对值 abs(x)	Integer、real	real	返回 x 的绝对值
	平方 sqr(x)			返回 x 的平方
	平方根 sqrt(x)			返回 x 的平方根
	正弦 sin(x)			返回 x 的正弦
	余弦 cos(x)			返回 x 的余弦
	反正切 arctan(x)			返回 x 的反正切
	指数 exp(x)			返回以 e 为底的幂值
	对数 ln(x)			返回以 e 为底的 x 的自然对数
	最小数 min(x,y)			返回 x 和 y 的最小值
	最大数 max(x,y)			返回 x 和 y 的最大值
	圆周率 pi	无参数		返回圆周率
	随机数 random(n)	integer	integer	返回 0～n 之间的随机整数，如果没有参数则返回 0～1 之间的随机数
	类型上界值 high(i)	顺序类型 顺序类型名	同参数	返回参数类型的最大值
	类型下界值 low(i)			返回参数类型的最小值
	数组下标上界 high(i)	数组变量	integer	返回数组下标的最大值
	数组下标下界 low(i)			返回数组下标的最小值
转换函数	序数 ord(x)	有序类型(整型、字符型、布尔型、枚举型、子界型)	integer	取序号，字符类取值 ASCII 值

续表

分 类	函 数 名	参数类型	结果类型	说 明
转换函数	字符 chr(i)	整型	字符型	返回 ASCII 值为 i 的字符
	截尾 trunc(x)	实型	整型	去掉 x 的小数部分取整
	舍入 round(x)	实型	整型	四舍五入取整
顺序函数	前趋 pred(i)	有序类型(整型、字符型、布尔型、枚举型、子界型)	同参数	返回 i 的前趋值
	后继 succ(i)			返回 i 的后继值

表 2 字符串运算的函数与过程

函数/过程名	说 明
AppendStr(s,s1)	在 s 后面添加字符串 s1
Concat(s1,s2)	连接两个或多个字符串
Copy(s,i,n)	返回 s 从第 i 个开始的 n 个字符组成的字符串
CompareStr(s1,s2)	比较两个字符串 s1 和 s2。若 s1>s2 则返回值大于 0,若 s1<s2 则返回值小于 0,若 s1=s2 则返回值为 0
Comparetext(s1,s2)	作用同上,但是不分大小写
Delete(s,i,n)	删除 s 中从第 i 个开始的 n 个字符
IfThen(b,s1,s2)	如果 b=True,则返回字符串 s1,否则返回字符串 s2
Insert(s,s1,n)	在 s 中的第 n 个位置插入字符串 s1
LeftStr(s,n)	返回 s 中左边 n 个字符形成的字符串
RightStr(s,n)	返回 s 中右边 n 个字符形成的字符串
Length(s)	返回字符串 s 的长度
Lowercase(s)	将字符串 s 中的所有英文字母转化为大写字母,其他字符保持不变
MidStr(s,p,n)	返回字符串 s 中从第 p 个开始的 n 个字符形成的字符串
Pos(s1,s)	返回字符串 s1 在 s 中的位置,若 s1 不在 s 中则返回 0
Upcase(c)	将字符 c 变成大写字符
Uppercase(s)	将字符串 s 中的所有英文字母转化为大写字母,其他字符保持不变
Str(x,s)	将数值 x 转化为字符串 s
Inttostr(n)	将整数 n 转化为字符类型
Floattostr(x)	将实数 x 转化为字符类型
Strtoint(s)	将字符串 s 转化为整数
Strtofloat(s)	将字符串 s 转化为实数

续表

函数/过程名	说　明
Trim(s)	除掉 s 左右不可见字符
TrimLeft(s)	除掉 s 左边不可见字符
TrimRight(s)	除掉 s 右边不可见字符
Date	返回当前日期
Time	返回当前时间
Now	返回当前日期和时间
Datetostr(date)	将日期 date 转化为字符串
Timetostr(time)	将时间 time 转化为字符串
Datetimetostr(datetime)	将日期时间 datetime 转化为字符串
Encodedate(year,month,day)	将年(year)、月(month)和日(day)转化为日期类型
Decodedate(date,year,month,day)	将日期类型 date 分解为年(year)、月(month)和日(day)
Encodetime(hour,min,sec)	将时(hour)、分(min)和秒(sec)转化为时间类型
Decodetime(time,hour,min,sec)	将时间类型 time 分解为时(hour)、分(min)和秒(sec)
Dayofweek(date)	返回 date 对应的星期数。例如,星期天为 1,星期一为 2,以此类推

计算机精品学习资料大放送

软考官方指定教材及同步辅导书下载 | 软考历年真是解析与答案

软考视频 | 考试机构 | 考试时间安排

Java 一览无余：Java 视频教程 | Java SE | Java EE

.Net 技术精品资料下载汇总：ASP.NET 篇

.Net 技术精品资料下载汇总：C#语言篇

.Net 技术精品资料下载汇总：VB.NET 篇

撼世出击：C/C++编程语言学习资料尽收眼底 电子书+视频教程

Visual C++(VC/MFC)学习电子书及开发工具下载

Perl/CGI 脚本语言编程学习资源下载地址大全

Python 语言编程学习资料(电子书+视频教程)下载汇总

最新最全 Ruby、Ruby on Rails 精品电子书等学习资料下载

数据库精品学习资源汇总：MySQL 篇 | SQL Server 篇 | Oracle 篇

最强 HTML/xHTML、CSS 精品学习资料下载汇总

最新 JavaScript、Ajax 典藏级学习资料下载分类汇总

网络最强 PHP 开发工具+电子书+视频教程等资料下载汇总

UML 学习电子资下载汇总 软件设计与开发人员必备

经典 LinuxCBT 视频教程系列 Linux 快速学习视频教程一帖通

天罗地网：精品 Linux 学习资料大收集(电子书+视频教程) Linux 参考资源大系

Linux 系统管理员必备参考资料下载汇总

Linux shell、内核及系统编程精品资料下载汇总

UNIX 操作系统精品学习资料<电子书+视频>分类总汇

FreeBSD/OpenBSD/NetBSD 精品学习资源索引 含书籍+视频

Solaris/OpenSolaris 电子书、视频等精华资料下载索引